

INTERNATIONAL STANDARD

IEC
61158-5

Second edition
2000-01

**Digital data communications for
measurement and control –
Fieldbus for use in industrial control systems –**

**Part 5:
Application Layer Service definition**



Reference number
IEC 61158-5:2000(E)

Numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series.

Consolidated publications

Consolidated versions of some IEC publications including amendments are available. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Validity of this publication

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology.

Information relating to the date of the reconfirmation of the publication is available in the IEC catalogue.

Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is to be found at the following IEC sources:

- **IEC web site***
- **Catalogue of IEC publications**
Published yearly with regular updates
(On-line catalogue)*
- **IEC Bulletin**
Available both at the IEC web site* and as a printed periodical

Terminology, graphical and letter symbols

For general terminology, readers are referred to IEC 60050: *International Electrotechnical Vocabulary (IEV)*.

For graphical symbols, and letter symbols and signs approved by the IEC for general use, readers are referred to publications IEC 60027: *Letter symbols to be used in electrical technology*, IEC 60417: *Graphical symbols for use on equipment. Index, survey and compilation of the single sheets* and IEC 60617: *Graphical symbols for diagrams*.

* See web site address on title page.

INTERNATIONAL STANDARD

IEC
61158-5

Second edition
2000-01

Digital data communications for measurement and control – Fieldbus for use in industrial control systems –

Part 5: Application Layer Service definition

© IEC 2000 – Copyright – all rights reserved

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

International Electrotechnical Commission
Telefax: +41 22 919 0300

3, rue de Varembé Geneva, Switzerland
e-mail: inmail@iec.ch IEC web site <http://www.iec.ch>



Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE **XH**

For price, see current catalogue

CONTENTS

Page

FOREWORD	9
INTRODUCTION	11
Clause	
1 SCOPE	11
2 NORMATIVE REFERENCES	12
3 DEFINITIONS	13
3.1 ISO/IEC 7498-1 terms	13
3.2 ISO/IEC 8822 terms	13
3.3 ISO/IEC 9545 terms	13
3.4 ISO/IEC 8824 terms	13
3.5 Fieldbus Data Link Layer terms	13
3.6 FAL Specific Definitions	14
3.6.1 Access Protection Definition	14
3.6.2 Address Assignment Table Definition	14
3.6.3 Allocate Definition	14
3.6.4 Application Layer Interoperability Definition	14
3.6.5 Application Object Definition	14
3.6.6 Application Process Definition	14
3.6.7 Application Process Identifier Definition	14
3.6.8 Application Process Object Class Definition	15
3.6.9 Application Relationship Definition	15
3.6.10 Application Relationship Application Service Element Definition	15
3.6.11 Application Relationship Endpoint Definition	15
3.6.12 Application Definition	15
3.6.13 Application Objects Definition	15
3.6.14 Attribute Definition	15
3.6.15 Behaviour Definition	15
3.6.16 Bit-no Definition	15
3.6.17 Channel Definition	15
3.6.18 Channel Related Diagnosis Definition	15
3.6.19 Class Definition	15
3.6.20 Class Attributes Definition	16
3.6.21 Class Code Definition	16
3.6.22 Class Specific Service Definition	16
3.6.23 Client Definition	16
3.6.24 Communication Objects Definition	16
3.6.25 Connection Definition	16
3.6.26 Connection ID (CID) Definition	16
3.6.27 Connection Path Definition	16
3.6.28 Connection Point Definition	16
3.6.29 Consume Definition	16
3.6.30 Consumer Definition	16
3.6.31 Consuming Application Definition	16
3.6.32 Configuration Check Definition	17
3.6.33 Configuration Fault Definition	17
3.6.34 Configuration Identifier Definition	17
3.6.35 Control Commands Definition	17
3.6.36 Conveyance Path Definition	17
3.6.37 Cyclic Definition	17
3.6.38 Data Consistency Definition	17

3.6.39	Dedicated AR Definition	17
3.6.40	Default DL-Address Definition.....	17
3.6.41	Device Definition	17
3.6.42	Device Profile Definition	17
3.6.43	Diagnosis Information Definition	17
3.6.44	Diagnosis Information Collection Definition	17
3.6.45	DP-Master (Class 1) Definition	18
3.6.46	DP-Master (Class 2) Definition	18
3.6.47	DP-Slave Definition	18
3.6.48	Dynamic AR Definition	18
3.6.49	End Node Definition	18
3.6.50	Endpoint Definition	18
3.6.51	Error Definition	18
3.6.52	Error Class Definition	18
3.6.53	Error Code Definition	18
3.6.54	FAL Subnet Definition	18
3.6.55	First-In-First-Out Variable Definition	18
3.6.56	Frame Definition.....	18
3.6.57	Freeze Definition	18
3.6.58	Group Definition	19
3.6.59	Invocation Definition.....	19
3.6.60	I/O Data Definition.....	19
3.6.61	Ident Number Definition	19
3.6.62	Identifier Related Diagnosis Definition	19
3.6.63	Index Definition	19
3.6.64	Instance Definition	19
3.6.65	Instance Attributes Definition	19
3.6.66	Instantiated Definition	19
3.6.67	Lpacket Definition	19
3.6.68	Management Information Definition.....	19
3.6.69	Manufacturer ID Definition	19
3.6.70	Master Parameter Set Definition.....	20
3.6.71	Member Definition.....	20
3.6.72	Message Router Definition.....	20
3.6.73	Multicast Connection Definition.....	20
3.6.74	Module Definition	20
3.6.75	Network Definition.....	20
3.6.76	Object Definition.....	20
3.6.77	Object Specific Service Definition	20
3.6.78	Originator Definition	20
3.6.79	Peer Definition	20
3.6.80	Point-to-point Connection Definition	20
3.6.81	Pre-defined AR Endpoint Definition	21
3.6.82	Pre-established AR Endpoint Definition.....	21
3.6.83	Process Data Definition	21
3.6.84	Produce Definition.....	21
3.6.85	Producer Definition	21
3.6.86	Publisher Definition	21
3.6.87	Publishing Manager Definition	21
3.6.88	Pull Publisher Definition	21
3.6.89	Pull Publishing Manager Definition	21
3.6.90	Push Publisher Definition.....	21
3.6.91	Push Publishing Manager Definition	21
3.6.92	Pull Subscriber Definition.....	22
3.6.93	Push Subscriber Definition.....	22
3.6.94	Real Configuration Definition	22
3.6.95	Resource Definition.....	22
3.6.96	Route Endpoint	22
3.6.97	Serial Number Definition	22
3.6.98	Server Definition	22
3.6.99	Server Definition	22

3.6.100	Service Definition	22
3.6.101	Slot Definition	22
3.6.102	Subscriber Definition	22
3.6.103	Sync Definition	22
3.6.104	Target Definition	23
3.6.105	Unconnected Message Manager (UCMM) Definition	23
3.6.106	Unconnected Service Definition	23
3.7	Abbreviations and Symbols	23
3.8	Conventions	24
3.8.1	General Conventions	24
3.8.2	Conventions for Class Definitions	25
3.8.3	Conventions for Service Definitions	26
4	CONCEPTS	28
4.1	Architectural relationships	28
4.1.1	Overview	28
4.1.2	Relationship to the Application Layer of the OSI Reference Model	28
4.1.3	Relationships to other fieldbus standards	29
4.2	Fieldbus Application Layer structure	30
4.2.1	Overview	30
4.2.2	Fundamental concepts	30
4.2.3	Fieldbus Application processes	31
4.2.4	Application objects	34
4.2.5	Application entities	36
4.2.6	Fieldbus Application Service Elements	37
4.2.7	Application relationships	40
4.3	Fieldbus Application Layer Naming and Addressing	43
4.3.1	Identifying Objects accessed through the FAL	43
4.3.2	Addressing APs accessed through the FAL	44
4.4	Architecture Summary	44
4.5	FAL Service Procedures and Time Sequence Diagrams	45
4.5.1	FAL Confirmed Service Procedures	45
4.5.2	Confirmed Service Time Sequence Diagram	46
4.5.3	FAL Unconfirmed Service Procedures	47
4.5.4	Unconfirmed Service Time Sequence Diagram	47
4.6	Common FAL attributes	47
4.7	Common FAL Service Parameters	48
4.8	APDU size	50
5	DATA TYPE ASE	51
5.1	Overview	51
5.1.1	Overview Of Basic Types	52
5.1.2	Overview of Constructed Types	52
5.1.3	Specification of User Defined Data Types	53
5.1.4	Transfer of User Data	53
5.2	Formal Definition of Data Type Objects	53
5.2.1	Data Type Class	53
5.3	FAL Defined Data Types	55
5.3.1	Fixed Length Types	55
5.3.2	String Types	69
5.3.3	Structure Types	71
5.4	Data type ASE Service Specification	73
5.5	Summary of data types	73
6	TYPES OF COMMUNICATION MODELS	74
7	COMMUNICATION MODEL TYPE 1 SPECIFICATION	74
7.1	Type 1 Concepts	74
7.2	Type 1 ASEs	74
7.2.1	Object Management ASE	74
7.2.2	Application Process ASE	84

7.2.3	Application Relationship ASE.....	108
7.2.4	Variable ASE	133
7.2.5	Event ASE	163
7.2.6	Load Region ASE.....	192
7.2.7	Function Invocation ASE.....	214
7.3	Type 1 ARs	232
7.3.1	Queued User-triggered Uni-directional (QUU) AR Endpoint Formal Model.....	232
7.3.2	Queued User-triggered Bi-directional Connection-Oriented (QUB-Co) AR Endpoint Formal Model	233
7.3.3	Queued User-triggered Bi-directional Connectionless (QUB-CL) AR Endpoint Formal Model.....	235
7.3.4	Queued User-triggered Bi-directional with Flow Control (QUB-FC) AR Endpoint Formal Model.....	238
7.3.5	Queued User-Triggered Bi-directional with Segmentation(QUB-Seg) AR Endpoint Formal Model	241
7.3.6	Buffered User-triggered Bi-directional (BUB) AR Endpoint Formal Model.....	244
7.3.7	Buffered Network-Scheduled Uni-directional (BNU) AR Endpoint Formal Model.....	247
7.3.8	Buffered Network-scheduled Bi-directional (BNB) AR Endpoint Formal Model.....	251
7.3.9	Buffered Network-Scheduled And Unscheduled Uni-directional (BNU-Mp) AR Endpoint Formal Model	254
7.4	Description.....	254
L.1	Formal Class Definition	257
7.5	Summary of Type 1 FAL Classes.....	259
7.6	Permitted FAL Type 1 Services by AREP Role	260
8	COMMUNICATION MODEL TYPE 2 SPECIFICATION	262
8.1	Type 2 Concepts.....	262
8.1.1	General	262
8.1.2	General concepts	262
8.1.3	Relationships between ASE's	263
8.1.4	Naming and Addressing.....	264
8.1.5	Type 2 Data Types	264
8.2	Type 2 ASEs.....	269
8.2.1	Type 2 Object Management ASE	269
8.2.2	Connection Manager ASE.....	317
8.3	Type 2 AR's	328
8.3.1	Overview	328
8.3.2	UCMM AR Formal Model	338
8.3.3	Transport AR Formal Model.....	340
8.3.4	AR ASE Services	357
8.4	Summary of FAL Type 2 Classes.....	365
8.5	Permitted FAL Type 2 Services by AR Type	366
9	COMMUNICATION MODEL TYPE 3 SPECIFICATION	367
9.1	Type 3 Concepts.....	367
9.1.1	User Requirements of the Type 3 DP System	367
9.1.2	Features of Type 3 DP	367
9.1.3	DP Communication Model	368
9.1.4	Dynamical Behaviour of Type 3 DP	379
9.1.5	Type 3 Status Values	380
9.2	Type 3 ASEs.....	381
9.2.1	Process Data ASE	381
9.2.2	I/O Data ASE.....	392
9.2.3	Diagnosis ASE	412
9.2.4	Alarm ASE.....	432
9.2.5	Context ASE.....	439
9.2.6	Management ASE	472
9.2.7	AR ASE	493
9.3	Type 3 ARs	498
9.3.1	MS0 Application Relationship.....	498
9.3.2	MS1 Application Relationship.....	501

9.3.3	MS2 Application Relationship.....	502
9.3.4	MM1 Application Relationship.....	506
9.3.5	MM2 Application Relationship.....	507
9.4	Summary of Type 3 FAL Classes.....	598
9.5	Permitted FAL Type 3 Services by AREP Role.....	599
9.6	Conformance Classes.....	601
9.7	Application Characteristics.....	602
9.7.1	Time Behaviour.....	602
9.7.2	Ident Number.....	603
10	COMMUNICATION MODEL TYPE 4 SPECIFICATION	603
10.1	Type 4 Concepts.....	603
10.1.1	Overview.....	603
10.1.2	Application Entities.....	603
10.1.3	Gateway and Routing.....	605
10.1.4	Architecture Summary.....	606
	FAL Service Procedures and Time Sequence Diagrams.....	607
10.2	Type 4 Variable ASE.....	609
10.2.1	Overview.....	609
10.2.2	Variable Model Class Specification.....	611
10.2.3	Basic Variable Type Specifications.....	612
10.2.4	Constructed Variable Type Specifications.....	617
10.2.5	Services.....	621
10.3	Type 4 Route Endpoint ASE.....	621
10.3.1	Overview.....	621
10.3.2	Route Endpoint Model.....	622
10.3.3	Route Endpoint ASE Service Specification.....	624
10.4	Type 4 Application Relationship ASE.....	629
10.4.1	Overview.....	629
10.4.2	Application Relationship Class Specification.....	630
10.4.3	Application Relationship ASE Service Specifications.....	632
11	COMMUNICATION MODEL TYPE 5 SPECIFICATION	636
11.1	Type 5 Concepts.....	636
11.1.1	Objectives of the Type 5 FAL AE.....	636
11.1.2	Type 5 Devices.....	636
11.1.3	Data Type ASE.....	637
11.1.4	APO ASEs.....	637
11.1.5	Application Relationships.....	639
11.1.6	Support for Configuration Management.....	640
11.2	Type 5 ASEs.....	640
11.2.1	Simple Virtual Field Device ASE.....	640
11.2.2	System Management Kernel ASE.....	647
11.2.3	LAN Redundancy ASE.....	664
11.3	Type 5 ARs.....	668
11.3.1	Publisher/Subscriber Type 5 AR Endpoint Formal Model.....	668
11.3.2	Client / Server AR Endpoint Formal Model.....	669
11.4	Summary of FAL Type 1 and Type 5 Classes.....	671
11.5	Permitted FAL Type 1 and Type 5 Services by AREP Role.....	672
12	COMMUNICATION MODEL TYPE 6 SPECIFICATION	673
13	COMMUNICATION MODEL TYPE 7 SPECIFICATION	673
13.1	Type 7 Concepts.....	673
13.1.1	AL Type 7 environment.....	673
13.1.2	Application services.....	673
13.1.3	Application process.....	674
13.1.4	Application entity.....	675
13.1.5	AE model.....	676
13.1.6	AEI model.....	677
13.1.7	Application context.....	679

13.1.8	Coordination of the AEs	683
13.1.9	Concepts for Sub_MMS	687
13.2	Type 7 ASEs	689
13.2.1	MPS ASE (Periodic/Aperiodic Manufacturing Services)	689
13.2.2	Virtual Model of a Device (VMD) ASE	783
13.2.3	Domain ASE	803
13.2.4	Program Invocation (PI) ASE	820
13.2.5	Variable ASE	833
13.2.6	Event ASE	851
13.2.7	Directory ASE	864
13.3	Type 7 ARs	864
13.3.1	Messaging Common Service (MCS) AR ASE	864
14	COMMUNICATION MODEL TYPE 8 SPECIFICATION	885
14.1	Type 8 Concepts	885
14.1.1	Overview	885
14.1.2	Mapping of functional capabilities	885
14.1.3	Fundamental concept restrictions	885
14.1.4	Common FAL Attributes	887
14.2	Type 8 ASEs	887
14.2.1	Overview	887
14.2.2	Object Management ASE	887
14.2.3	Application Process ASE	888
14.2.4	Application Relationship ASE	892
14.2.5	Variable ASE	895
14.2.6	Function Invocation ASE	899
14.3	Type 8 Application Relationships	903
14.3.1	Queued User-triggered Bi-directional with Flow Control (QUB-FC) AR Endpoint Formal Model	903
14.1.1	Buffered Network-Scheduled Uni-directional (BNU) AR Endpoint Formal Model	903
14.2	Permitted FAL Services by AREP Role	904
Annex A. (informative)	Model for Service Error Reporting	905
A.1	General	905
A.2	Error Handling Procedure	906
Bibliography		908

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**DIGITAL DATA COMMUNICATIONS FOR MEASUREMENT AND CONTROL –
FIELDBUS FOR USE IN INDUSTRIAL CONTROL SYSTEMS –**

Part 5: Application Layer Service definition

FOREWORD

- 1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international cooperation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. 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. The 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 the 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 National Committees.
- 3) The documents published have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
- 4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
- 5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.
- 6) Attention is drawn to the possibility that some of the elements of this standards may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

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

This second edition cancels and replaces the first edition which was issued as a technical specification in 1999. It constitutes a technical revision and now has the status of an International Standard.

This second edition adds seven distinct sets of services, each with a corresponding protocol, to the set of services and protocols of the first edition.

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

FDIS	Report on voting
65C/225/FDIS	65C/230/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 the ISO/IEC Directives, Part 3.

IEC 61158 consists of the following parts, under the general title *Digital data communications for measurement and control — Fieldbus for use in industrial control systems*:

- Part 1: Introductory guide (under preparation)
- Part 2: Physical layer specification and service definition
- Part 3: Data Link Service definition
- Part 4: Data Link Protocol specification
- Part 5: Application layer service definition
- Part 6: Application layer protocol specification
- Part 7: System management (under consideration)
- Part 8: Conformance testing (under consideration)

Annexe A is for information only.

The committee has decided that the contents of this publication will remain unchanged until 2006. At this date, the publication will be:

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

INTRODUCTION

This standard describes the Fieldbus Application Layer services intended to support the information interchange and the interactions between application processes.

This application layer standard does not specify individual implementations or products, nor does it constrain the implementations of application entities and interfaces within the industrial automation system.

This application layer standard does not contain test specifications used to demonstrate compliance with IEC 61158-5 and IEC 61158-6.

IECNORM.COM: Click to view the full PDF of IEC 61158-5:2000

Withd2W

DIGITAL DATA COMMUNICATIONS FOR MEASUREMENT AND CONTROL – FIELDBUS FOR USE IN INDUSTRIAL CONTROL SYSTEMS –

Part 5: Application Layer Service definition

1 Scope

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”.

The FAL is an Application Layer Communication Standard designed to support the conveyance of time-critical and non-time-critical application requests and responses among devices in an automation environment. The term “time-critical” is used to represent the presence of an application time-window, within which one or more specified actions are required to be completed with some defined level of certainty.

This standard specifies the structure and services of the IEC Fieldbus Application Layer (FAL). It is specified 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.

This part of IEC 61158 specifies interactions between remote applications in terms of

- an abstract model for defining application resources (objects) capable of being manipulated by users via the use of FAL Services;
- the primitives (interactions between the FAL and the FAL user) associated with each FAL Service;
- the parameters associated with each primitive;
- the interrelationship between and the valid sequences of the primitives for each service.

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.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 61158. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

IEC 61131-1: 1992, *Programmable controllers – Part 1: General information*

IEC 61131-3:1993, *Programmable controllers — Part 3: Programming languages*

IEC 61158-3:2000, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems– Part 3: Data Link Layer service definition*

IEC 61158-4: 2000, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 4: Data Link Layer Protocol Specification*

IEC 61158-6: 2000, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 6: Application Layer Protocol Specification*

ISO/IEC 646: 1991, *Information technology – ISO 7-bit coded character set for information interchange*

ISO/IEC 7498 (all parts), *Information technology – Open Systems Interconnection – Basic Reference Model*

ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO 7498-3:1997, *Information technology – Open Systems Interconnection – Basic Reference Model – Part 3: Naming and addressing*

ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application Layer structure*

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

ANSI/IEEE 754:– *IEEE Standard for Binary Floating-Point arithmetic*

IEEE 754:1985 (R1990), *Binary floating-point arithmetic*

3 Definitions

For the purposes of this part of IEC 61158, the following definitions apply.

3.1 ISO/IEC 7498-1 terms

For the purposes of this part of IEC 61158, 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; and
- i) transfer syntax.

3.2 ISO/IEC 8822 terms

For the purposes of this part of IEC 61158, the following terms as defined in ISO/IEC 8822 apply:

- a) abstract syntax; and
- b) presentation context.

3.3 ISO/IEC 9545 terms

For the purposes of this part of IEC 61158, 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; and
- i) application control service element.

3.4 ISO/IEC 8824 terms

For the purposes of this part of IEC 61158, the following terms as defined in ISO/IEC 8824 apply:

- a) object identifier; and
- b) type.

3.5 Fieldbus Data Link Layer terms

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

- a) DL-Time
- b) DL-Scheduling-policy
- c) DLCEP
- d) DLC
- e) DL-connection-oriented mode
- f) DLSDU
- g) DLSAP
- h) link
- i) network address / node address / MAC ID
- j) node
- k) tag (fixed tag, generic tag)
- l) scheduled, unscheduled

3.6 FAL Specific Definitions

For the purposes of this part of IEC 61158, the following definitions apply.

3.6.1 Access Protection Definition

The access to certain functions of a server is restricted to one client.

3.6.2 Address Assignment Table Definition

Mapping of the client's internal I/O-addresses to the decentralised inputs and outputs.

3.6.3 Allocate Definition

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

3.6.4 Application Layer Interoperability Definition

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

3.6.5 Application Object Definition

component of an application process that is identifiable and accessible through an FAL application relationship. Application 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 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 objects and their corresponding definitions

3.6.6 Application Process Definition

Part of a distributed application on a network, which is located on one device and unambiguously addressed.

3.6.7 Application Process Identifier Definition

If multiple application processes are used in a device these application processes are addressed with an Application Process Identifier.

3.6.8 Application Process Object Class Definition

description of a class of application process objects in terms of the set of its network-accessible attributes and services.

3.6.9 Application Relationship Definition

cooperative relationship between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation. This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities

3.6.10 Application Relationship Application Service Element Definition

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

3.6.11 Application Relationship Endpoint Definition

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

3.6.12 Application Definition

function or data structure for which data is consumed or produced.

3.6.13 Application Objects Definition

multiple object classes that manage and provide the run time exchange of messages across the network and within the network device.

3.6.14 Attribute Definition

a description of an externally visible characteristic or feature of an object. 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 behaviour of an object. Attributes are divided into class attributes and instance attributes.

3.6.15 Behaviour Definition

indication of how the object responds to particular events. Its description includes the relationship between attribute values and services.

3.6.16 Bit-no Definition

A Bit-no designates the number of a bit in a bitstring or an octet.

3.6.17 Channel Definition

A single physical or logical connection of inputs or outputs of a server to the process.

3.6.18 Channel Related Diagnosis Definition

This diagnosis information refers to a single input or output channel of the server.

3.6.19 Class Definition

a set of objects, all of which represent the same kind of system component. A class is a generalisation of the object; a template for defining variables and methods. All objects in a class are identical in form and behaviour, but usually contain different data in their attributes.

3.6.20 Class Attributes Definition

an attribute that is shared by all objects within the same class.

3.6.21 Class Code Definition

a unique identifier assigned to each object class.

3.6.22 Class Specific Service Definition

a service defined by a particular object class to perform a required function which is not performed by a common service. A class specific object is unique to the object class which defines it.

3.6.23 Client Definition

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

(2) an initiator of a message to which a server reacts, such as the role of an AR endpoint in which it issues confirmed service request APDUs to a single AR endpoint acting as a server

3.6.24 Communication Objects Definition

components that manage and provide run time exchange of messages across the network such as the Connection Manager object, the unconnected message manager (UCMM), and the Message Router object.

3.6.25 Connection Definition

a logical binding between two application objects. These application objects may be within the same or different devices.

3.6.26 Connection ID (CID) Definition

identifier assigned to a transmission that is associated with a particular connection between producers and consumers that identifies a specific piece of application information.

3.6.27 Connection Path Definition

the attribute is made up of a byte stream which defines the application object to which a connection instance applies.

3.6.28 Connection Point Definition

a buffer which is represented as a subinstance of an assembly object.

3.6.29 Consume Definition

the act of receiving data from a producer.

3.6.30 Consumer Definition

a node that is receiving data from a producer.

3.6.31 Consuming Application Definition

the application that consumes data.

3.6.32 Configuration Check Definition

In the start-up phase, the client transmits the expected configuration for comparison with the real configuration to the server.

3.6.33 Configuration Fault Definition

If the server detects a not acceptable difference between the expected configuration and the real configuration, then a configuration fault is indicated in the diagnosis information.

3.6.34 Configuration Identifier Definition

For each input- and/or output-module of a server at least one Configuration Identifier is defined in the configuration string (either a Cfg Identifier or Special Cfg Identifier).

3.6.35 Control Commands Definition

Control commands will be transferred from the client to the server. With the control commands it is possible to clear the outputs, or to freeze the inputs and/or synchronise the outputs of the server.

3.6.36 Conveyance Path Definition

unidirectional flow of APDUs across an application relationship

3.6.37 Cyclic Definition

term used to describe events which repeat in a regular and repetitive manner.

3.6.38 Data Consistency Definition

The input- or output-data-range shall always be transmitted consistent between the client and the server.

3.6.39 Dedicated AR Definition

AR used directly by the FAL User. On Dedicated ARs, only the FAL Header and the user data are transferred

3.6.40 Default DL-Address Definition

The default DL-address of a server is 126. This address will be used as initial value if address setting will be done via Type 3 DP.

3.6.41 Device Definition

a physical hardware connection to the link. A device may contain more than one node.

3.6.42 Device Profile Definition

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

3.6.43 Diagnosis Information Definition

This term is used to identify the whole diagnosis data of a server.

3.6.44 Diagnosis Information Collection Definition

A system diagnosis information that is collected at the client.

3.6.45 DP-Master (Class 1) Definition

The DP-Master (Class 1) is a controlling device which controls several DP-Slaves (field devices); usually a programmable controller or distributed control system.

3.6.46 DP-Master (Class 2) Definition

The DP-Master (Class 2) is a controlling device which manages configuration data (parameter sets) and diagnosis data of a DP-Master (Class 1). Additionally the DP-Master (Class 2) can perform all communication capabilities of a DP-Master (Class 1).

3.6.47 DP-Slave Definition

A field device that is assigned to one DP-Master (Class 1) as a provider for cyclic I/O data exchange. In addition acyclic functions and alarms could be provided.

3.6.48 Dynamic AR Definition

AR that requires the use of the AR establishment procedures to place it into an established state

3.6.49 End Node Definition

a producing or consuming node.

3.6.50 Endpoint Definition

one of the communicating entities involved in a connection.

3.6.51 Error Definition

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

3.6.52 Error Class Definition

general class for error definitions. Error codes for specific errors are defined within an error class

3.6.53 Error Code Definition

identification of a specific type of error within an error class

3.6.54 FAL Subnet Definition

networks composed of one or more data link segments. They are permitted to contain bridges, but not routers. FAL subnets are identified by a subset of the network address.

3.6.55 First-In-First-Out Variable Definition

A FIFO (First In First Out) is a Type 4 Variable Object class, composed of a set of homogeneously typed elements. The first written element will be the first element that can be read. On the Fieldbus only one, complete element can be transferred as a result of one service invocation.

3.6.56 Frame Definition

single data transfer on a link.

3.6.57 Freeze Definition

The input data of several DP-Slaves will be simultaneously frozen.

3.6.58 Group Definition

A set of DP-Slaves which perform a Freeze or Sync operation.

3.6.59 Invocation Definition

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

3.6.60 I/O Data Definition

Data which have to be transferred cyclically for the purpose of processing.

3.6.61 Ident Number Definition

The Ident Number specifies a DP-Master (Class 1) or DP-Slave device type.

3.6.62 Identifier Related Diagnosis Definition

This diagnosis information is related to the input- or output-range for which an Identifier is defined in the configuration string.

3.6.63 Index Definition

Address of an object within an application process.

3.6.64 Instance Definition

the actual physical occurrence of an object within a class. Identifies one of many objects within the same object class. For example: California is an instance of the object class state. The terms object, instance, and object instance are used to refer to a specific instance.

3.6.65 Instance Attributes Definition

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

3.6.66 Instantiated Definition

an object that has been created in a device.

3.6.67 Lpacket Definition

the type 2 Lpacket (or link packet) is a piece of application information that contains a size, control byte, tag, and link data. Peer Data Link Layers use Lpackets to send and receive service data units from higher layers in the OSI stack.

3.6.68 Management Information Definition

network-accessible information that supports the management of the Fieldbus environment, including the application layer

3.6.69 Manufacturer ID Definition

identification of each manufacturer of Type 2 products by number. Manufacturer IDs uniquely identify a particular manufacturer of Type 2 products.

3.6.70 Master Parameter Set Definition

The Master parameter set includes the configuration and parameterisation data of all DP-Slaves that are assigned to the corresponding DP-Master and the bus parameters.

3.6.71 Member Definition

a piece of an attribute that is structured as an array.

3.6.72 Message Router Definition

the object within a node that distributes messaging requests to the appropriate application objects.

3.6.73 Multicast Connection Definition

a connection from one node to many. Multicast connections allow messages from a single producer to be received by many consumer nodes.

3.6.74 Module Definition

Addressable unit inside the DP-Slave.

3.6.75 Network Definition

a series of nodes connected by some type of communication medium. The connection paths between any pair of nodes can include repeaters, routers and gateways.

3.6.76 Object Definition

an abstract representation of a particular component within a device, i.e. :

- (1) An abstract representation of a computer's capabilities. Objects can be composed of any or all of the following components:
 - a) data (information which changes with time),
 - b) configuration (parameters for behaviour),
 - c) methods (things that can be done using data and configuration).
- (2) A collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behaviour.

3.6.77 Object Specific Service Definition

a service defined by a particular object class to perform a required function which is not performed by a common service. An object specific service is unique to the object class which defines it.

3.6.78 Originator Definition

the client responsible for establishing a connection path to the target.

3.6.79 Peer Definition

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

3.6.80 Point-to-point Connection Definition

a connection that exists between two nodes only. Connections can be either point-to-point or multicast.

3.6.81 Pre-defined AR Endpoint Definition

AR endpoint that is defined locally within a device without use of the create service. Pre-defined ARs that are not pre-established must be established before being used

3.6.82 Pre-established AR Endpoint Definition

AR endpoint that is placed in an established state during configuration of the AEs that control its endpoints

3.6.83 Process Data Definition

Data which are already pre-processed and transferred acyclically for the purpose of information or further processing.

3.6.84 Produce Definition

act of sending data to be received by a consumer.

3.6.85 Producer Definition

a node that is responsible for sending data.

3.6.86 Publisher Definition

role of an AR endpoint in which it transmits APDUs onto the Fieldbus for consumption by one or more subscribers. The publisher may not be aware of the identity or the number of subscribers and it may publish its APDUs using a dedicated AR. Two types of publishers are defined by this standard, Pull Publishers and Push Publishers, each of which is defined separately

3.6.87 Publishing Manager Definition

role of an AR endpoint in which it issues one or more confirmed service request APDUs to a Publisher to request the Publisher to publish a specified object. Two types of publishing managers are defined by this standard, Pull Publishing Managers and Push Publishing Managers, each of which is defined separately

3.6.88 Pull Publisher Definition

type of Publisher that publishes an object in confirmed service response APDUs that are responses to confirmed service request APDUs received from its Pull Publishing Manager. The AR used to support Pull Publishers conveys the response APDUs to one or more Pull Subscriber AR endpoints

3.6.89 Pull Publishing Manager Definition

type of Publishing Manager that sends confirmed service request APDUs to its Publisher to request that the specified object be published in the corresponding response APDU

3.6.90 Push Publisher Definition

type of Publisher that publishes an object in unconfirmed service request APDUs. These unconfirmed service request APDUs may be issued as a result of a confirmed service request APDU received on a separate AREP from its Push Publishing Manager. The AR used to support Push Publishers conveys the response APDUs to one or more Pull Subscriber AR endpoints.

3.6.91 Push Publishing Manager Definition

type of Publishing Manager that sends confirmed service requests to its Publisher to request that the specified object be published using an unconfirmed service on a separate Push Publisher AR endpoint

3.6.92 Pull Subscriber Definition

type of Subscriber that recognizes received confirmed service response APDUs as published object data

3.6.93 Push Subscriber Definition

type of Subscriber that recognizes received unconfirmed service request APDUs as published object data

3.6.94 Real Configuration Definition

The real configuration describes the input and output data structure of the DP-Slave. Additionally, the definition of the data consistency is included in the real configuration.

3.6.95 Resource Definition

resource is a processing or information capability of a subsystem

3.6.96 Route Endpoint

A Route Endpoint is a Type 4 object container, containing Variable Objects of a Type 4 variable class.

3.6.97 Serial Number Definition

a unique value (32 bit integer) assigned by each manufacturer to every device having Type 2 communication capabilities. Each manufacturer shall be responsible for guaranteeing the uniqueness of the serial number across all of its devices, but the number need only be unique with respect to the manufacturer. When used in conjunction with the Manufacturer ID, it shall form a unique identifier for each device.

3.6.98 Server Definition

role of an AREP in which it returns a confirmed service response APDU to the Client that initiated the request

3.6.99 Server Definition

an object which provides services to another (client) object.

3.6.100 Service Definition

operation or function that an object and/or object class performs upon request from another object and/or object class. A set of common services is defined and provisions for the definition of object-specific services are provided. Object-specific services are those which are defined by a particular object class to perform a required function which is not performed by a common service.

3.6.101 Slot Definition

Address of a module within a DP-Slave.

3.6.102 Subscriber Definition

role of an AREP in which it receives APDUs produced by a publisher. Two types of subscribers are defined by this standard, Pull Subscribers and Push Subscribers, each of which is defined separately

3.6.103 Sync Definition

The outputs of several DP-Slaves will be simultaneously updated.

3.6.104 Target Definition

the end-node to which a connection is established.

3.6.105 Unconnected Message Manager (UCMM) Definition

the component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object.

3.6.106 Unconnected Service Definition

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

3.7 Abbreviations and Symbols

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
APO	Application 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	The Control and Information Protocol corresponding to Type 2. CIP includes both connected and unconnected messaging.
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
DLCEP	Data Link Connection End Point
DLL	Data Link Layer
DLM	Data Link-management
DLSAP	Data Link Service Access Point
DLSDU	DL-service-data-unit
DP	Decentralised Peripherals
FAL	Fieldbus Application Layer

FIFO	First In First Out
ID	Identifier
IEC	International Electrotechnical Commission
Ind	Indication
ISO	International Organization for Standardization
LME	Layer Management Entity
O2T	Originator to target (connection characteristics)
O⇒T	Originator to target (connection characteristics)
OSI	Open Systems Interconnect
PDU	Protocol Data Unit
PL	Physical Layer
PLC	Programmable Logic Controller
REP	Route Endpoint
Req	Request
Rsp	Response
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)
T⇒O	Target to originator (connection characteristics)
VAO	Type 4 Variable Object

3.8 Conventions

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate clause. Each ASE specification is composed of two parts, its class specification, and its service specification. The protocol specification for each of the ASEs is defined in IEC 61158-6.

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

3.8.1 General Conventions

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

3.8.2 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 2048 are allocated for identifying user defined classes.
- (4) The "PARENT CLASS:" entry is the name of the parent class for the class being specified. All attributes defined for the parent class and inherited by it are inherited for the class being defined, and therefore do not have to be redefined in the template for this class.

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

- (5) The "ATTRIBUTES" label indicate that the following entries are attributes defined for the class.
- a) Each of the attribute entries contains a line number in column 1, a mandatory (m) / optional (o) / conditional (c) / selector (s) indicator in column 2, an attribute type label in column 3, a name or a conditional expression in column 4, and optionally a list of enumerated values in column 5. In the column following the list of values, the default value for the attribute may be specified.
 - b) Objects are normally identified by a numeric identifier or by an object name, or by both. In the class templates, these key attributes are defined under the key attribute.
 - c) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting is used to specify
 - i) fields of a structured attribute (4.1,4.2, 4.3),
 - ii) attributes conditional on a constraint statement (5). Attributes may be mandatory (5.1) or optional (5.2) if the constraint is true. Not all optional attributes require constraint statements as does the attribute defined in (5.2).
 - iii) the selection fields of a choice type attribute (6.1 and 6.2).
- (6) The "SERVICES" label indicates that the following entries are services defined for the class.
- a) An (m) in column 2 indicates that the service is mandatory for the class, while an (o) indicates that it is optional. A (c) in this column indicates that the service is conditional. When all services defined for a class are defined as optional, at least one has to be selected when an instance of the class is defined.
 - b) The label "OpsService" designates an operational service (1).
 - c) The label "MgtService" designates an management service (2).
 - d) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting within the list of services is used to specify services conditional on a constraint statement.

3.8.3 Conventions for Service Definitions

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

3.8.3.1 Service Parameters

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

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

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

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

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

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

- a) a parameter-specific constraint:
 - “(=)” indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.
- b) an indication that some note applies to the entry:
 - “(n)” indicates that the following note "n" contains additional information pertaining to the parameter and its use.

3.8.3.2 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

This clause describes fundamentals of the FAL. Each of the ASEs specified for the FAL are described, including their relationships to each other. More detailed descriptive information about each of the FAL ASEs can be found in the “Overview” clause of their specifications.

4.1 Architectural relationships

4.1.1 Overview

The Fieldbus is intended to be used in factories and process plants to interconnect primary automation devices (e.g. sensors, actuators, local display devices, annunciators, programmable logic controllers, small single loop controllers, and stand-alone field controls) with control and monitoring equipment located in control rooms.

Primary automation devices are associated with the lowest levels of the industrial automation hierarchy and perform a limited set of functions within a definite time window. Some of these functions include diagnostics, data validation, and handling of multiple inputs and outputs.

These primary automation devices, also termed field devices, are located close to the process fluids, the fabricated part, the machine, the operator and the environment. This use positions the Fieldbus at the lowest levels of the Computer Integrated Manufacturing (CIM) architecture.

Some of the expected benefits in using Fieldbus are reduction in wiring, increase in amount of data exchanged, wider distribution of control between the primary automation devices and the control room equipment, and the satisfaction of time critical constraints.

4.1.2 Relationship to the Application Layer of the OSI Reference Model

The functions of the FAL have been described according to OSI layering principles. However, its architectural relationship to the lower layers is different, as shown in Figure 1.

- The FAL includes OSI functions together with extensions to cover time-critical requirements. The OSI Application Layer Structure standard (ISO/IEC 9545) was used as a basis for specifying the FAL.
- The Fieldbus Data Link layer directly supports the FAL.

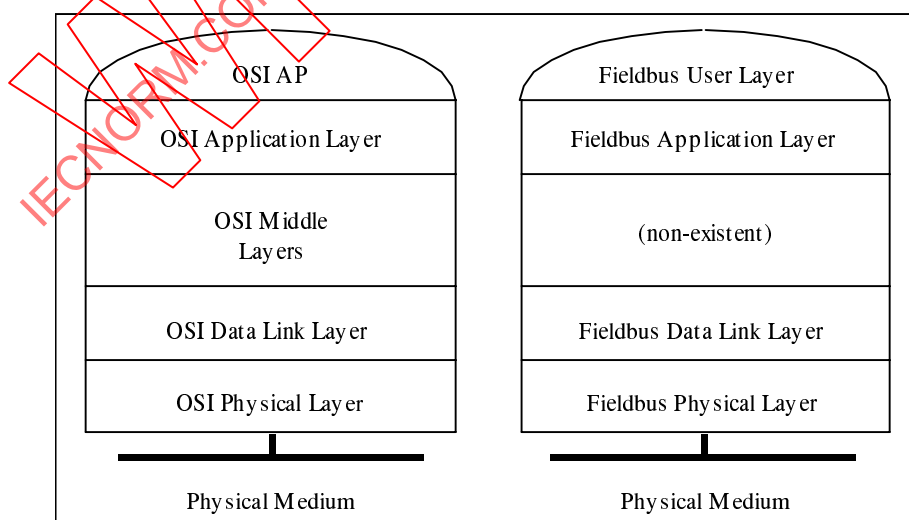


Figure 1 – Relationship to the ISO OSI Reference Model

4.1.3 Relationships to other fieldbus standards

The Fieldbus Application Layer (FAL) architectural relationships, as illustrated in Figure 2, have been designed to support the interoperability needs of time-critical systems distributed within the Fieldbus environment.

Within this environment, the FAL provides communications services to time-critical and non-time-critical applications located in fieldbus devices.

In addition, the FAL directly uses the Data Link Layer to transfer its application layer protocol data units. It does this using a set of data transfer services and a set of supporting services used to control the operational aspects of the Data Link Layer.

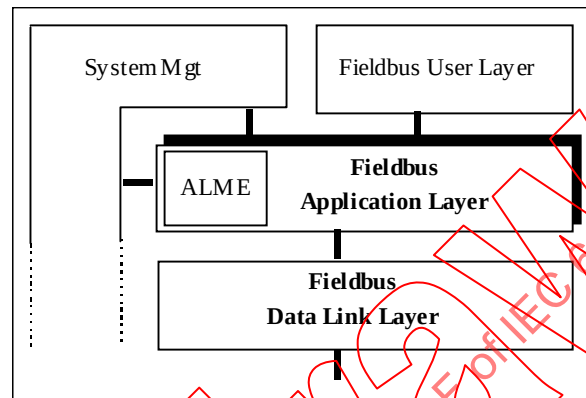


Figure 2 – Architectural Positioning of the Fieldbus Application Layer

4.1.3.1 Use of the Fieldbus Data Link Layer

The Fieldbus Application Layer (FAL) provides network access to Fieldbus APs. It interfaces directly to the Fieldbus Data Link Layer for transfer of its APDUs.

The Data Link Layer provides various types of services to the FAL for the transfer of data between Data Link Service access points (DLSAPs) and Data Link connection endpoints (DLCEPs). The Data Link Layer protocol includes scheduled and non-scheduled access to the physical medium.

4.1.3.2 Support for Fieldbus applications

Fieldbus applications are represented to the network as application processes (APs). APs are the components of a distributed system that may be individually identified and addressed.

Each AP contains an FAL application entity (AE) that provides network access for the AP. That is, each AP communicates with other APs through its AE. In this sense, the AE provides a window of visibility into the AP.

APs contain identifiable components that are also visible across the network. These components are represented to the network as Application Process Objects (APO). They may be identified by one or more key attributes. They are located at the address of the application process that contains them.

The services used to access them are provided by APO-specific application service elements (ASEs) contained within the FAL. These ASEs are designed to support user, function block, and management applications.

4.1.3.3 Support for System Management

The FAL services can be used to support various management operations, including management of fieldbus systems, applications, and the Fieldbus network.

NOTE System management will be specified in the future IEC 61158-7 (in preparation).

4.1.3.4 Access to FAL Layer Management Entities

One layer management entity (LME) is present in each FAL entity on the network. FALMEs provide access to the FAL for system management purposes.

The set of data accessible by the System Manager is referred to as the System Management Information Base (SMIB). Each Fieldbus Application Layer Management Entity (FALME) provides the FAL portion of the SMIB. How the SMIB is implemented is beyond the scope of this standard.

4.2 Fieldbus Application Layer structure

4.2.1 Overview

The structure of the FAL is a refinement of the OSI Application Layer Structure (ISO/IEC 9545). As a result, the organization of this subclause is similar to that of ISO/IEC 9545. Certain concepts presented here have been refined from ISO/IEC 9545 for the Fieldbus environment.

The FAL differs from the other layers of OSI in two principal aspects:

- OSI defines a single type of application layer communications channel, the association, to connect APs to each other. The FAL defines the Application Relationship (AR), of which there are several types, to permit application processes (APs) to communicate with each other.
- The FAL uses the DLL to transfer its PDUs and not the presentation layer. Therefore, there is no explicit presentation context available to the FAL. Between the same pair (or set) of data link service access points the FAL protocol may not be used concurrently with other application layer protocols.

4.2.2 Fundamental concepts

The operation of time critical real open systems is modeled in terms of interactions between time-critical APs. The FAL permits these APs to pass commands and data between them.

Cooperation between APs requires that they share sufficient information to interact and carry out processing activities in a coordinated manner. Their activities may be restricted to a single Fieldbus segment, or they may span multiple segments. The FAL has been designed using a modular architecture to support the messaging requirements of these applications.

Cooperation between APs also often requires that they share a common sense of time. The Data Link Layer specified in IEC 61158-3 and IEC 61158-4 provides for the distribution of time to all devices. It also defines local device services that can be used by APs to access the distributed data link time.

The remainder of this clause describes each of the modular components of the architecture and their relationships with each other. The components of the FAL are modeled as objects, each of which provides a set of FAL communication services for use by applications. The FAL objects and their relationships are described below. The detailed specifications of FAL objects and their services are provided in the following clauses of this standard. IEC 61158-6 specifies the protocols necessary to convey these object services between applications.

4.2.3 Fieldbus Application processes

4.2.3.1 Definition of the Fieldbus AP

In the Fieldbus environment, an application may be partitioned into a set of components and distributed across a number of devices on the network. Each of these components is referred to as a Fieldbus Application Process (AP). A Fieldbus AP is a variation of an Application Process as defined in ISO OSI Reference Model (ISO/IEC 7498). Fieldbus APs may be unambiguously addressed by at least one individual Data Link Layer service access point address. Unambiguously addressed, in this context, means that no other AP may simultaneously be located by the same address. This definition does not prohibit an AP from being located by more than one individual or group data link service access point address.

4.2.3.2 Communication services

Fieldbus APs communicate with each using confirmed and unconfirmed services (ISO/IEC 10731). The services defined in this standard for the FAL specify the semantics of the services as seen by the requesting and responding APs. The syntax of the messages used to convey the service requests and responses is defined in IEC 61158-6. The AP behavior associated with the services is specified by the AP.

Confirmed services are used to define request/response exchanges between APs, such that if a response is not returned to the requesting user within a specified time interval, the requesting user is notified and the service invocation is cancelled.

Unconfirmed services, in contrast, are used to define the unidirectional transfer of messages from one AP to one or more remote APs. From a communications perspective, there is no relationship between separate invocations of unconfirmed services as there is between the request and response of a confirmed service. Therefore, there is no associated time interval associated with the processing of an unconfirmed service by the receiver.

Dedicated ARs provide the AR ASE services directly to the FAL User. They support optimized data transfers between APs in which user data is transferred using only one byte FAL Protocol Control Information (PCI).

4.2.3.3 AP Interactions

Within the Fieldbus environment, APs may interact with other APs as necessary to achieve their functional objectives. No constraints are imposed by this standard on the organization of these interactions or the possible relationships that may exist between them. For example, in the Fieldbus environment, interactions may be based on request/response messages sent directly between APs, or on data/events published by one AP for use by others. These two models of interactions between APs are referred to as client/server and publisher/subscriber interactions.

The services supported by an interaction model are conveyed by application relationship endpoints (AREPs) associated with the communicating APs. The role that the AREP plays in the interaction (e.g. client, server, peer, publisher, subscriber) is defined as an attribute of the AREP. See clause 7 for the definition of the AREP and its attributes. See the annexes following annex B for the definition of each of the AREP types, including their roles. See annex C for a summary of AREP Roles and the FAL services supported by them.

4.2.3.3.1 Client/Server Interactions

Client/server interactions are characterized by a bi-directional data flow between a single client AP and a single server AP, as shown in Figure 3. In this type of interaction, the client may issue a confirmed or unconfirmed request to the server to perform some task. If the service is confirmed then the server will always return a response. If the service is unconfirmed, the server may return a response using an unconfirmed service defined for this purpose.

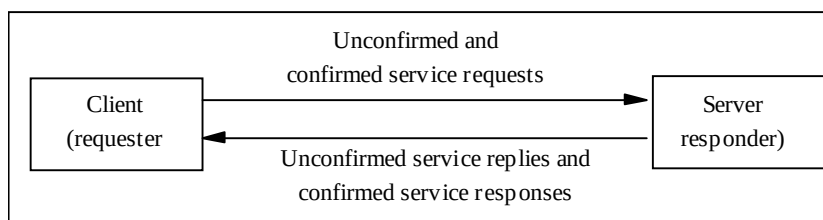


Figure 3 – Client/Server Interactions

4.2.3.3.2 Publisher / Subscriber Interactions

Publisher/subscriber interactions, on the other hand, involve a single publisher AP, and a group of one or more subscriber APs. This type of interaction has been defined to support two models of interaction between APs, the "pull" model and the "push" model. In both models, the setup of the publishing AP is performed by management and is outside the scope of this standard.

4.2.3.3.2.1 Pull Model Interactions

In the "pull" model, the publisher receives a request to publish from a remote publishing *manager*, and broadcasts (or multicasts) its response across the network. The publishing manager is responsible only for initiating publishing by sending a request to the publisher.

Subscribers wishing to receive the published data listen for responses transmitted by the publisher. In this fashion, data is "pulled" from the publisher by requests from the publishing manager.

Confirmed FAL services are used to support this type of interaction. Two characteristics of this type of interaction differentiate it from the other types of interaction. First, a typical confirmed request/response exchange is performed between publishing manager and the publisher. However, the underlying conveyance mechanism provided by the FAL returns the response not just to the publishing manager, but also to all subscribers wishing to receive the published information. This is accomplished by having the Data Link Layer transmit the response to a group address, rather than to the individual address of the publishing manager. Therefore, the response sent by the publisher contains the published data and is multicast to the publishing manager and to all subscribers.

The second difference occurs in the behavior of the subscribers. Pull model subscribers, referred to as pull subscribers, are capable of accepting published data in confirmed service responses without having issued the corresponding request. Figure 4 illustrates these concepts.

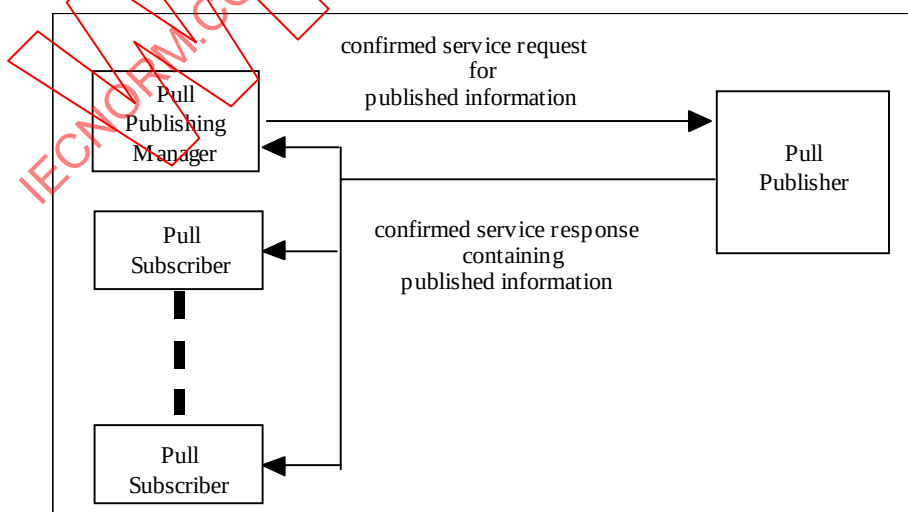


Figure 4 – Pull Model Interactions

4.2.3.3.2 Push Model Interactions

In the "push" model, two services are used, one confirmed and one unconfirmed. The confirmed service is used by the subscriber to request to join the publishing. The response to this request is returned to the subscriber, following the client/server model of interaction. This exchange is only necessary when the subscriber and the publisher are located in different APs.

The unconfirmed service used in the Push Model is used by the publisher to distribute its information to subscribers. In this case, the publisher is responsible for invoking the correct unconfirmed service at the appropriate time and for supplying the appropriate information. In this fashion, it is configured to "push" its data onto the network.

Subscribers for the Push Model receive the published unconfirmed services distributed by publishers. Figure 5 illustrates the concept of the Push Model.

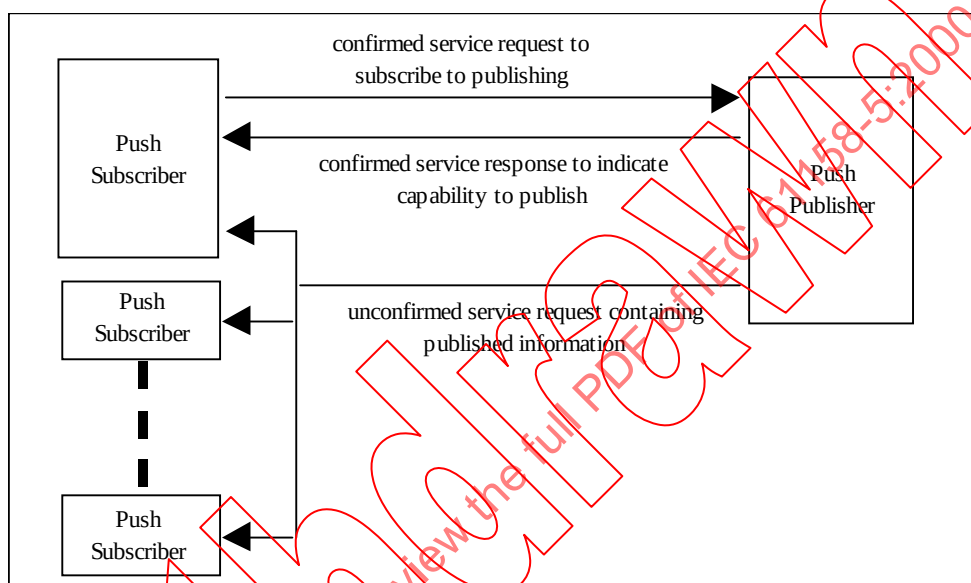


Figure 5 – Push Model Interactions

4.2.3.3.2.3 Timeliness of published information

To support the perishable nature of published information, the FAL supports four types of timeliness defined for publisher/subscriber interactions. Each make it possible for subscribers of published data to determine if the data they are receiving is up-to-date or "stale". These types are realized through mechanisms within the Data Link Layer (DLL). Each is described briefly below. For a more detailed description, refer to IEC 61158-3 and IEC 61158-4.

<u>Type</u>	<u>Description</u>
Transparent	This type of timeliness allows the user application process to determine the timeliness quality of data that it generates and have the timeliness quality accompany the information when it is transferred across the network. In this type of timeliness, the network provides no computation or measurement of timeliness. It merely conveys the timeliness quality provided with the data by the user application process.
Residence	When the FAL submits data from the publishing AP to the DLL for transmission, the DLL starts a timer. If the timer expires before the data has been transmitted, the DLL marks the buffer as "not timely" and conveys this timeliness information with the data.
Synchronized	This type of timeliness requires the coordination of two pieces of published information. One is the data to be published and the other is a special "sync mark". When the sync mark is received from the network a timer starts in each of the participating stations. Subsequently, when data is received for

transmission by the DLL at the publishing station, or when the transmitted data is received from the network at a subscribing station, the DLL timeliness attribute for the data is set to TRUE. It remains TRUE until the reception of the next sync mark or until the timer expires. Data received after the timer expires but before the next sync mark does not cause the timeliness attribute to be reset to TRUE. It is only reset to TRUE if data is received within the time window after receipt of the sync mark. Data transmitted by the publisher station with the timeliness attribute set to FALSE maintains the setting of FALSE at each of the subscribers, regardless of their timer operation.

Update This type of timeliness requires the coordination of the same two pieces of published information defined for *synchronized* timeliness. In this type, the sync mark also starts a timer in each of the participating stations. Like *synchronized* timeliness, expiration of the timer always causes the timeliness attribute to be set to FALSE. Unlike *synchronized* timeliness, receipt of new data at any time (not just within the time window started with the receipt of a sync mark) causes the timeliness attribute to be set to TRUE.

NOTE Time interval values are managed through system management (see the future IEC 61158-7).

4.2.3.4 AP Structure

The internals of APs may be represented by one, or more Application Process Objects (APOs) and accessed through one or more Application Entities (AEs). AEs provide the communication capabilities of the AP. For each Fieldbus AP, there is one and only one FAL AE. APOs are the network representation of application specific capabilities (user application objects) of an AP that are accessible through its AEs.

4.2.3.5 AP Class

An AP class is a definition of the attributes and services of an AP. The standard class definition for APs is defined in this standard. User defined classes also may be specified. Class identifiers (described in 3.8.2) are assigned from a set reserved for this purpose.

4.2.3.6 AP Type

AP types provide the mechanism for defining standard APs.

As described above in the previous subclauses, APs are defined by instantiating an AP class. Each AP definition is composed of the attributes and services selected for the AP from those defined by its AP class. In addition, an AP definition contains values for one or more of the attributes selected for it. When two APs share the same definition, that definition is referred to as an AP type. Thus, an AP type is a generic specification of an AP that may be used to define one or more APs.

The descriptor reference attribute included in AP class definitions may be used to identify the type of the AP.

4.2.4 Application objects

4.2.4.1 Definition of APO

An application object (APO) is a network representation of a specific aspect of an AP. Each APO represents a specific set of information and processing capabilities of an AP that are accessible through services of the FAL. APOs are used to represent these capabilities to other APs in a Fieldbus system.

From the perspective of the FAL, an APO is modeled as a network accessible object contained within an AP or within another APO (APOs may contain other APOs). APOs provide the network definition for objects contained within an AP that are remotely accessible. The definition of an APO includes an identification of the FAL services that can be used by remote APs for remote access. The FAL services, as shown in Figure 6, are provided by the FAL communications entity of the AP, known as the FAL Applications Entity (FAL AE).

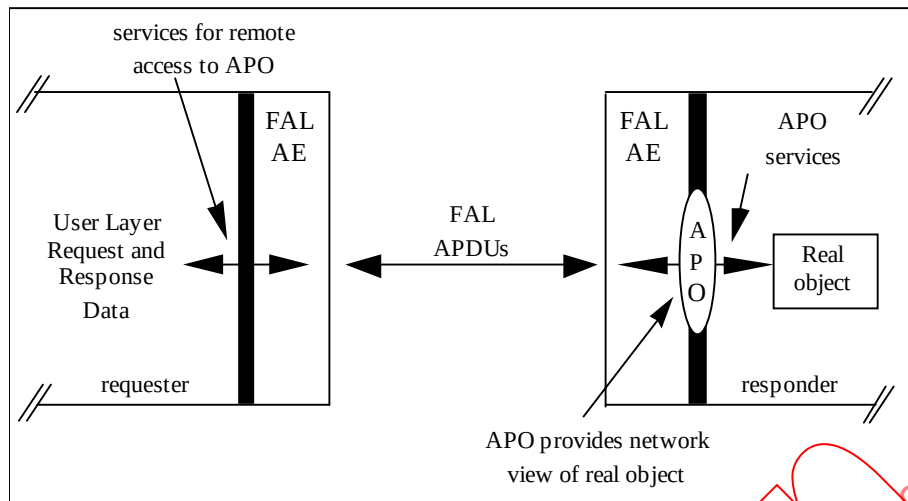


Figure 6 – APOs Services Conveyed by the FAL

In Figure 6, remote APs acting as clients 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.

To support the publisher / subscriber model of interaction, information about the real object can be published through its APO. Remote APs acting as subscribers see the APO view of the published information instead of having to know any of the real object specific details.

4.2.4.2 APO Classes

An APO Class is a generic specification for a set of APOs, each of which is described by the same set of attributes and accessed using the same set of services.

APO Classes provide the mechanism for standardizing network visible aspects of APs. Each standard APO Class definition specifies a particular set of network accessible AP attributes and services. IEC 61158-6 specifies the syntax and the procedures used by the FAL protocol to provide remote access to the attributes and services of an APO Class.

Standard APO Classes are specified by this standard for the purpose of standardizing remote access to APs. User defined classes may also be specified. Class identifiers for user defined classes, as described previously in 3.8.2, are assigned from a set reserved for this purpose.

User defined classes are defined as subclasses of standardized APO Classes or of other user-defined classes. They may be defined by identifying new attributes or by indicating that optional attributes for the parent class are mandatory for the subclass. The conventions for defining classes defined in 3.8.2 may be used for this purpose. The method for registering or otherwise making these new class definitions available for public use is beyond the scope of this standard.

4.2.4.3 APOs as Instances of APO Classes

APO Classes are defined in this standard using templates. These templates are used not only to define APO Classes, but also to specify the instances of a class.

Each APO defined for an AP is an instance of an APO Class. Each APO provides the network view of a real object contained in an AP. An APO is defined by

- (1) selecting the attributes from its APO Class template that are to be accessible from the real object,
- (2) assigning values to one or more attributes indicated as key in the template. Key attributes are used to identify the APO;
- (3) assigning values to zero, one, or more non-key attributes for the APO. Non-key attributes are used to characterize the APO;
- (4) selecting the services from the template that may be used by remote APs to access the real object.

Subclause 3.8.2 of this standard specifies the conventions for class templates. These conventions provide for the definition of mandatory, optional, and conditional attributes and services.

Mandatory attributes and services are required to be present in all APOs of the class. Optional attributes and services may be selected, on an APO by APO basis, for inclusion in an APO. Conditional attributes and services are defined with an accompanying constraint statement. Constraint statements specify the conditions that indicate whether or not the attribute is to be present in an APO.

4.2.4.4 APO Types

APO types provide the mechanism for defining standard APOs.

As described above in the previous subclauses, APOs are defined by instantiating an APO class. Each APO definition is composed of the attributes and services selected for the APO from those defined by its APO class. In addition, an APO definition contain values for one or more of the attributes selected for it. When two APOs share the same definition, except for the key attribute settings, that definition is referred to as an APO type. Thus, an APO type is a generic specification of an APO that may used to define one or more APOs.

The descriptor reference attribute included in APO Class definitions may be used to identify the type of the APO.

4.2.5 Application entities

4.2.5.1 Definition of FAL AE

An application entity provides the communication capabilities for a single AP. An FAL AE provides a set of services and the supporting protocols to enable communications between APs in a Fieldbus environment. The services provided by FAL AEs are grouped into Application Service Elements (ASE), such that the FAL services provided to an AP are defined by the ASEs its FAL AE contains. Figure 7 illustrates this concept.

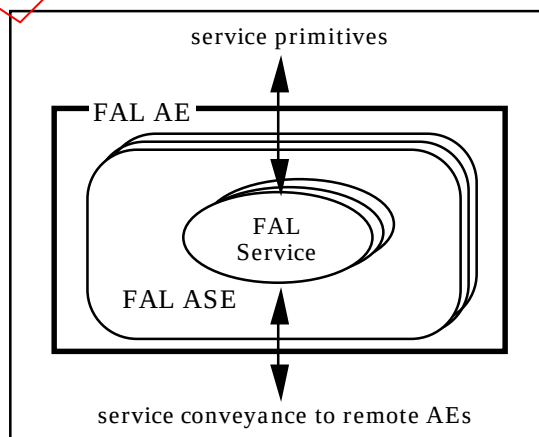


Figure 7 – Application entity structure

4.2.5.2 AE Type

Application entities that provide the same set of ASEs are of the same AE-type. Two AEs must share a common set of ASEs to be capable of communicating with each other.

4.2.6 Fieldbus Application Service Elements

4.2.6.1 General

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-entity-invocations for a specific purpose. ASEs provide a set of services for conveying requests and responses to and from application processes and their objects. AEs, as defined above, are represented by a collection of ASE invocations within the AE.

4.2.6.2 FAL Services

FAL Services convey functional requests/responses between APs. Each FAL service is defined to convey requests and responses for access to a real object modeled as an FAL accessible object.

The FAL defines both confirmed and unconfirmed services. Confirmed service requests are sent to the AP containing the real object. Each invocation of a confirmed service request is identified by a user supplied Invoke ID. The same Invoke ID is returned in the response by the AP containing the real object. It is used by the requesting AP and its FAL AE to associate the response with the appropriate request.

The FAL AE uses context information associated with the request to help decode the response. If the responding AP fails to respond within a specified time period, the FAL AE uses the Invoke ID to indicate to the requesting AP that the response has timed out and the FAL AE will no longer wait for it.

Unconfirmed services may be sent from the AP containing the real object to publish information about the object. They also may be sent to the AP containing the real object to access the real object. Both types of unconfirmed services are defined for the FAL.

4.2.6.3 Definition of FAL ASEs

A modular approach has been taken in the definition of FAL ASEs. The ASEs defined for the FAL are also object-oriented. Each ASE, except for the Object Management ASE, provides a set of services designed for one specific object class or for a related set of classes. The Object Management ASE provides a common set of management services applicable to all classes of objects. The AP ASE provides a set of services that operates on the AP itself. The Variable, Event, Function Invocation, and Load Region ASEs provide services for accessing the APOs of an AP.

To support remote access to the AP, the Application Relationship ASE is defined. It provides services to the AP for defining and establishing communication relationships with other APs, and it provides services to the other ASEs for conveying their service requests and responses.

Each FAL ASE defines a set of services, APDUs, and procedures that operate on the classes that it represents. Only a subset of the ASE services may be provided to meet the needs of an application. Profiles may be used to define such subsets. Definition of profiles is beyond the scope of this standard.

APDUs are sent and received between FAL ASEs that support the same services. Each FAL AE contains, at a minimum, the AR ASE and at least one other ASE. Figure 8 illustrates a subset of the FAL ASEs and their architectural relationships. All APO ASEs follow this model.

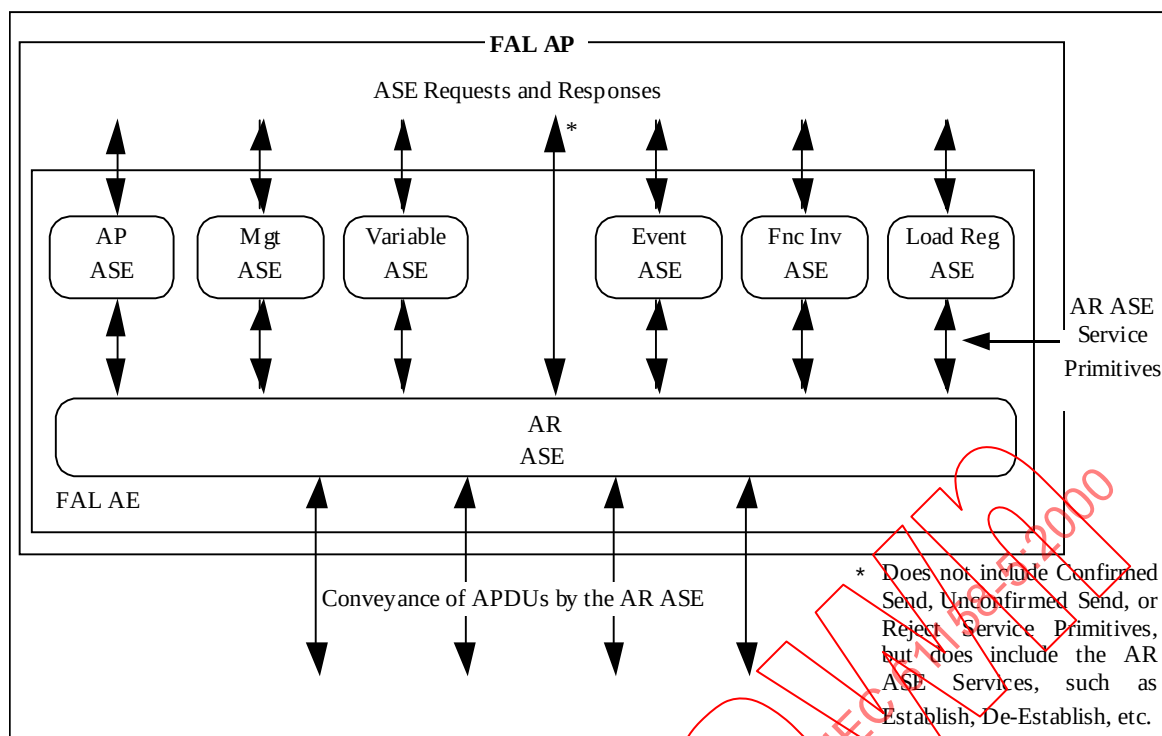


Figure 8 – FAL ASEs

4.2.6.3.1 Object Management ASE

A special object management ASE is specified for the FAL that provides services for the management of objects. Its services are used to access object attributes, and create and delete object instances. These services are used to manage network visible AP objects accessed through the FAL. The specific operational services that apply to each object type are specified in the definition of the ASE for the object type. Figure 9 illustrates the integration of management and operational services for an object within an AP.

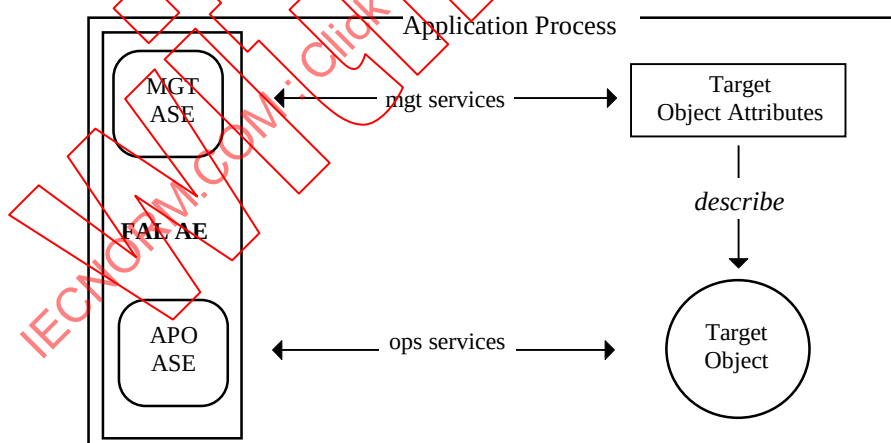


Figure 9 – FAL Management of objects

4.2.6.3.2 AP ASE

An AP ASE is specified for the identification and control of FAL APs. The attributes defined by the AP ASE specify characteristics of the AP about its manufacturer, and list its contents and capabilities. One service defined for the AP ASE supports the push model for publishing data. Using this service, subscribers are able to indicate their requirements for receiving published data. Another service is used to initialize communications with other APs. Other AP specific services are used to identify the manufacturer of the AP, to obtain its operational status.

4.2.6.3.3 APO ASEs

The FAL specifies a set of ASEs with services defined for accessing the APOs of an AP. The APO ASEs defined for the FAL are:

- Data Type ASE
- Variable ASE
- Event ASE
- Load Region ASE
- Function Invocation ASE

4.2.6.3.4 AR ASE

An AR ASE is specified to establish and maintain application relationships (ARs), that are used to convey FAL APDUs between/among APs. ARs represent application layer communication channels between APs. AR ASEs are responsible for providing services at the endpoints of ARs. AR ASE services are defined for establishing, terminating, and aborting ARs, for conveying APDUs for the AE, and for indicating the local status of the AR to the user. In addition, local services are defined for accessing certain aspects of AR endpoints.

Because ARs are communications objects defined and located within the FAL, they are managed by the System Management Agent instead of being managed using the FAL Object Management ASE services. The System Management Agent and the AR managed objects will be defined in the future IEC 61158-7.

4.2.6.4 FAL Service Conveyance

FAL AP, Management, and User APO ASEs provide services to convey requests and responses between service users and real objects. For push publishing services, service requests are conveyed *from* the real object, instead of to it. For all others, service requests are conveyed to the real object and service responses are returned from it.

To accomplish the task of conveying service requests and responses, three types of activities for the sending user and three corresponding types for the receiving user are defined. At the sending user, they accept service requests and responses to be conveyed. Secondly, they select the type of FAL APDU that will be used to convey the request or response and encode the service parameters into its body portion. Then they submit the encoded APDU body to the AR ASE for conveyance.

At the receiving user, they receive encoded APDU bodies from the AR ASE. They decode the APDU bodies and extract the service parameters conveyed by them. To conclude the conveyance, they deliver the service request or response to the user. Figure 10 illustrates these concepts.

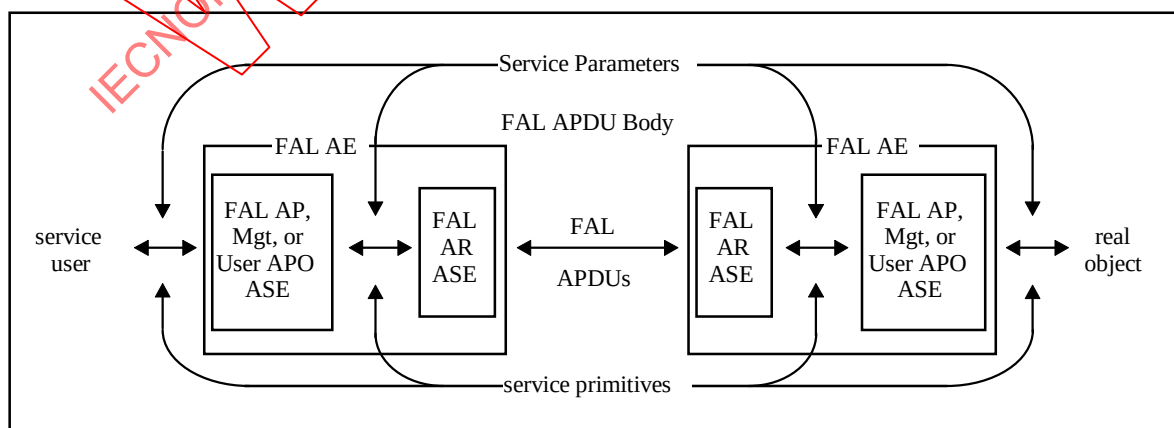


Figure 10 – ASE Service Conveyance

When a confirmed service request is received for conveyance the sending AR ASE starts a timer when it sends the request APDU and waits for the corresponding response APDU. If the timer expires before the response is received, it notifies the user and stops waiting for the response.

In addition, for service requests that operate on AR endpoints, such as establish and de-establish, the AR ASE will perform the processing required to maintain the appropriate context for the affected endpoint(s).

4.2.6.5 FAL Presentation Context

The presentation context in the OSI environment is used to distinguish the APDUs of one ASE from another, and to identify the transfer syntax rules used to encode each APDU. However, the Fieldbus communications architecture does not include the presentation layer. Therefore, an alternate mechanism is provided for the FAL by each of the specific types of Communication Models.

4.2.7 Application relationships

4.2.7.1 Definition of AR

ARs represent communication channels between APs. They define how information is communicated between APs. Each AR is characterized by how it conveys ASE service requests and responses from one AP to another. These characteristics are described below.

4.2.7.2 AR-Endpoints

ARs are defined as a set of cooperating APs. The AR ASE in each AP manages an endpoint of the AR, and maintains its local context. The local context of an AR endpoint is used by the AR ASE to control the conveyance of APDUs on the AR.

4.2.7.3 AR-Endpoint Classes

ARs are composed of a set of endpoints of compatible classes. AR endpoint classes are used to represent AR endpoints that convey APDUs in the same way. Through the standardization of endpoint classes, ARs for different models of interaction can be defined. Endpoint class definitions are contained as annexes to this standard. Additional classes may be added as necessary in annexes.

4.2.7.4 AR Cardinality

ARs characterize communications between APs. One of the characteristics of an AR is the number of AR endpoints in the AR. ARs that convey services between two APs have a cardinality of 1-to-1. Those that convey services from one AP to a number of APs have a cardinality of 1-to-many. ARs that represent a many-to-many conveyance among APs are not supported by the FAL.

4.2.7.5 Accessing Objects through ARs

ARs provide access to APs and the objects within them through the services of one or more ASEs. Therefore, one characteristic is the set of ASE services that may be conveyed to and from these objects by the AR. The list of services that can be conveyed by the AR are selected from those defined for the AE.

4.2.7.6 AR Conveyance Paths

ARs are modeled as one or two conveyance paths between AR endpoints. Each conveyance path conveys APDUs in one direction between one or more AR endpoints. Each receiving AR endpoint for a conveyance path receives all APDUs transmitting on the AR by the sending AR endpoint.

4.2.7.7 AREP Roles

Because APs interact with each other through endpoints, a basic determinant of their compatibility is the role that they play in the AR. The role defines how an AREP interacts with other AREPs in

the AR. Therefore, endpoint roles are defined to support the client/server and publisher/subscriber models of interaction.

For example, an AREP may operate as a client, a server, a publisher, or a subscriber. When an AREP interacts with another AREP on a single AR as both a client and a server, it is defined to have the role of “peer”. Other roles defined for AREPs are used in support of the publisher/subscriber model of interaction, such as publishing manager, “pull” publisher, and “push” publisher.

Certain roles are capable of initiating service requests, while others are only capable of responding to service requests. This part of the definition of a role identifies the requirement for an AR to be capable of conveying requests in either direction, or only in one direction.

4.2.7.8 AREP Buffers and Queues

AREPs may be modeled as a queue or as a buffer provided by the Data Link Layer. APDUs transferred over a queued AREP are delivered in the order received for conveyance. The transfer of APDUs over a buffered AREP is different. In this case, an APDU to be conveyed by the AR ASE is placed in a Data Link Layer buffer for transfer. When the Data Link Layer gains access to the network, it transmits the contents of the buffer.

When the AR ASE receives another conveyance request, it replaces the previous contents of the buffer whether or not they were transmitted. Once an APDU is written into a buffer for transfer, it is preserved in the buffer until the next APDU to be transmitted replaces it. While in the buffer, an APDU may be read more than once without deleting it from the buffer or changing its contents.

At the receiving end, the operation is similar. The receiving Data Link Layer places a received APDU into a buffer for access by the AR ASE. When a subsequent APDU is received, it overwrites the previous APDU in the buffer whether or not it was read by the AR ASE. Reading the APDU from the buffer is not destructive — it does not destroy or change the contents of the buffer, allowing the contents to be read from the buffer one or more times.

One variant of buffered ARs is that they can be dedicated to the FAL User, typically for the publishing of a single variable (or variable list). When they are, the identity of the variable (or variable list) is known by the context. ARs of this type are known as “dedicated”.

4.2.7.9 User-triggered and Network Scheduled Conveyance

Another characteristic of an AREP is when they convey service requests and responses. AREPs that convey them upon submission by the user are called user-triggered. Their conveyance is asynchronous with respect to network operation.

AREPs that convey requests and responses at predefined intervals, regardless of when they are received for transfer are termed network-scheduled. The time-intervals used to regulate conveyance are maintained by the Data Link Layer. Network-scheduled AREPs are capable of indicating when transferred data was submitted late for transmission, or when it was submitted on time, but transmitted late.

Network-scheduled AREPs may also convey requests and responses synchronously using the Data Link compel service if the Data Link layer has available bandwidth. These transfers occur “out-of-band” (that is, outside the cyclic portion of the network schedule) with respect to the scheduled transfers of these ARs.

4.2.7.10 AREP Timeliness

AREPs convey APDUs between applications using the services of the link layer. When the timeliness capabilities are defined for an AREP, the AREP forwards the timeliness indicators provided by the Data Link Layer. These timeliness indicators make it possible for subscribers of published data to determine if the data they are receiving is up-to-date or “stale”. A brief description of these timeliness indicators is given in 4.2.3.3.2.3.

To support these types of timeliness, the publishing AREP establishes a publisher data link connection reflecting the type of timeliness configured for it by management. After connection establishment, the AREP receives user data and submits it to the DLL for transmission, where timeliness procedures are performed. When the Data Link Layer has the opportunity to transmit the data, it transmits the current timeliness status with the data.

At the subscriber AREP, a data link connection is opened to receive published data that reflects the type of timeliness configured for it by management. The Data Link Layer computes the timeliness of received data and then delivers it to the AREP. The data is then delivered to the user AP through the appropriate ASE.

4.2.7.11 Identification of ARs

FAL AEs are accessed through one or more data link addresses. ARs are identified by the data link addresses of the AREPs that participate in the AR.

To simplify AR identification for 1-to-many ARs, group DLSAP addresses are used instead of address lists for AREPs that receive unconfirmed service requests transferred using connectionless data link services. For AREPs that receive unconfirmed service requests transferred using connection-oriented data link services, the DLCEP address of the sending AREP is used to identify the AR; no other address is necessary. Therefore, the addresses used to identify a 1-to-many AR are reduced to either a pair of DLSAP addresses or a single DLCEP address, instead of an enumerated list of addresses.

For the dissemination of information between publishers and subscribers, these techniques permit publishers to be unaware of the identity of the subscribers.

4.2.7.12 Definition and creation of AREPs

AREP definitions specify instances of AREP classes. AREPs may be predefined or they may be defined using the create service if their AE supports this capability.

AREPs may be pre-defined and pre-established, or they may be pre-defined and dynamically established. Figure 11 depicts these two cases. AREPs also may require both dynamic definition (through System Management) and establishment or they may be dynamically defined such that they may be used without any establishment (they are defined in an established state).

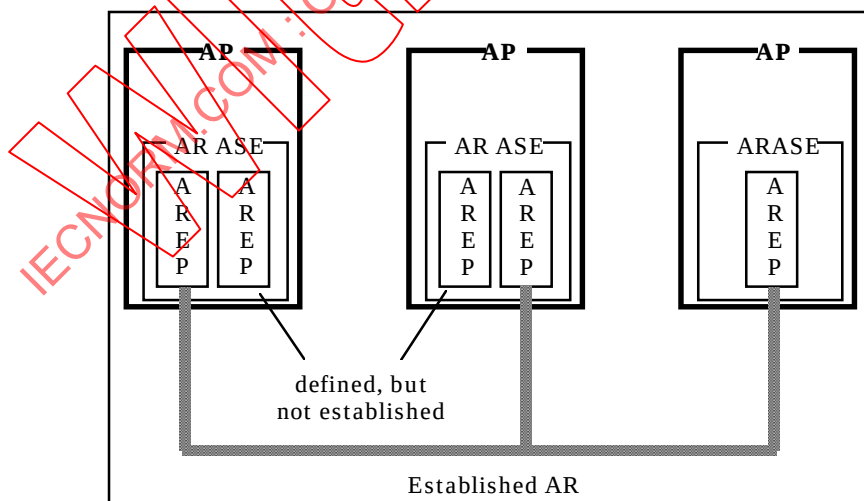


Figure 11 – Defined and Established AREPs

4.2.7.13 AR Establishment and Termination

ARs may be established either before the operational phase of the AP or during its operation. When established during the operation of an AP, the AR Establish service is used. The AR Establish service results in the exchange of AR Establishment APDUs.

Once an AR has been established, an AR may be terminated using the Abort service. The Abort service causes the AR endpoint to be closed immediately without the exchange of APDUs. Instead, the aborted endpoint issues a single abort APDU to the other endpoints participating in the AR that it is no longer capable of sending or receiving APDUs on the AR.

Additionally, an AR can be gracefully terminated, or de-established, using the AR DeEstablish service. The AR DeEstablish service results in the exchange of AR de-establishment APDUs. The definition of its operation is for future study.

4.2.7.14 Reserved ARs

The FAL defines a mechanism for specifying and reserving ARs for a specific purpose. One such AR is specified and reserved for use by system management. Reserved ARs are specified in IEC 61158-6, annex A.

4.3 Fieldbus Application Layer Naming and Addressing

This subclause refines the principles defined in ISO 7498-3 that involve the identification (naming) and location (addressing) of APOs referenced through the Fieldbus Application Layer.

This subclause defines how names and numeric identifiers are used to identify APOs accessible through the FAL. This subclause also specifies how Data Link Layer addresses are used to locate APs in the Fieldbus environment.

4.3.1 Identifying Objects accessed through the FAL

APOs accessed through the FAL are identified independent of their location. That is, if the location of the AP that contains the APO changes, the APO may still be referenced using the same set of identifiers.

Identifiers for APs and APOs within the FAL are defined as key attributes in the class definitions for APOs. Within these APO class definitions, four types of key attributes are used: names, numeric identifiers, symbolic addresses, and numeric addresses.

Identifiers are assigned for the purpose of discriminating among APs and APOs. Therefore, at a given time,

- (1) within the scope of a Fieldbus system, no two APs may have the same name, and
- (2) within the scope of an AP, no two APOs may have the same identifier, and
- (3) within the scope of an AP, an APO may not have the same identifier as its AP.

4.3.1.1 Names

Names are string-oriented identifiers. They are defined to permit APs and APOs to be named within the domain of the system where they are used. Therefore, although the scope of the name of an APO is specific to the AP in which it resides, the assignment of the name is administered within the system in which it is configured.

Names may be descriptive, although they do not have to be. Descriptive names make it possible to provide meaningful information, such as its use, about the object they name.

Names may also be coded. Coded names make it possible to identify an object using a short, compressed form of a name. They are typically simpler to transfer and process, but not as easy to understand as descriptive names.

The names used within the FAL may be either descriptive or coded. Their form and their values are assigned by the FAL user.

4.3.1.2 Numeric Identifiers

Numeric identifiers are identifiers whose values are integers. They are designed for efficient use within the Fieldbus system, and are assigned for efficient access to APOs by their AP. The scope of numeric identifiers is local to the AP.

4.3.1.3 Numeric Addresses

Numeric addresses represent logical memory addresses for APOs of a limited set of classes. This type of identifier is defined to simplify naming of APOs that are normally associated with a specific location in memory.

4.3.1.4 Symbolic Addresses

Symbolic addresses represent logical identifiers for APOs independent of their use or the system where they are configured. They are defined to permit manufacturers of Fieldbus APs to assign identifiers to APOs independent from names assigned within the domain of the system where they are used.

4.3.2 Addressing APs accessed through the FAL

Fieldbus addresses represent the network locations of APs. Addresses relevant to the FAL are DLSAP addresses used to locate the AREPs of an AP.

4.3.2.1 Use of Data Link Layer Addresses

DL-addresses are used by the FAL to locate AR endpoints. These addresses may be DLSAP-addresses or DLCEP-addresses, depending on how the AREP is mapped to the Data Link Layer. An AREP's Data Link Layer mapping defines which capabilities of the Data Link Layer are used to support conveyance of APDUs by the AREP.

DL addresses are not used to identify objects accessed or referenced through the AR endpoint, and they are only used to locate the endpoint, not identify it.

4.3.2.2 Binding of AR Endpoints to Data Link Layer Service Access Points

Data Link Layer addresses locate the Data Link Layer Service Access points (DLSAPs) and connection endpoints (DLCEPs) to which an AR endpoint may be bound. Therefore, an AR endpoint is located by its binding to a single DLSAP or DLCEP that may be identified by one or more DLL-addresses. Only one AR Endpoint may be bound to a specific DLCEP, or source and destination DLSAP pair in the case of connectionless data link operation, at a given time.

4.4 Architecture Summary

The summary of the FAL architecture is presented in this clause. Figure 12 illustrates the major components of the FAL architecture and how they relate to each other.

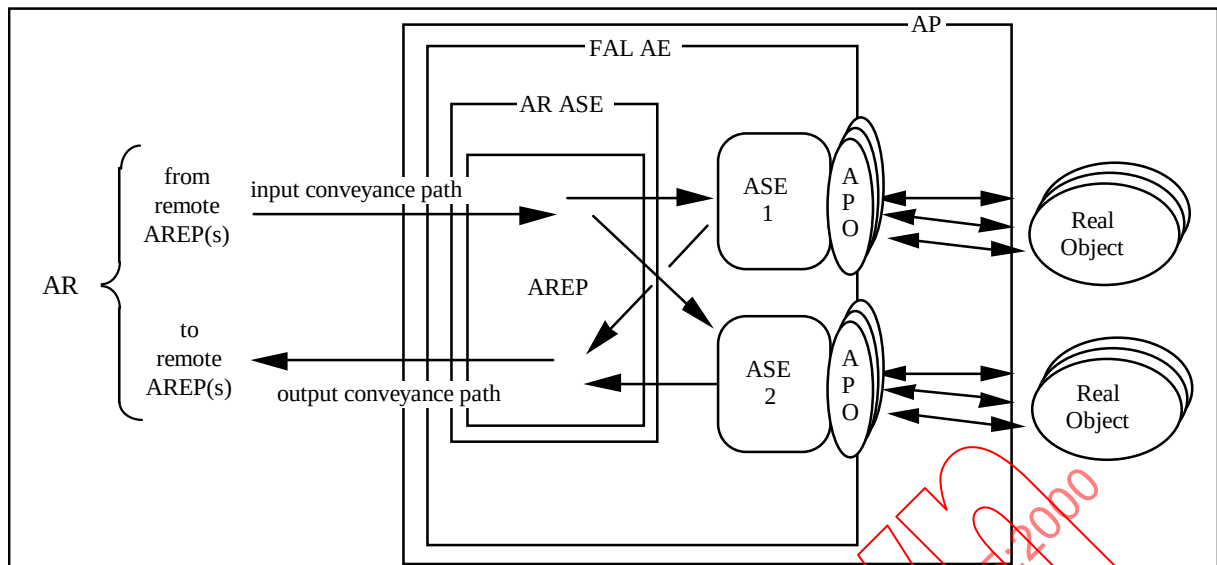


Figure 12 – FAL Architectural Components

Figure 12 depicts an AP that communicates through the FAL AE. The AP represents its internal real objects as APOs for remote access to them. Two ASEs are shown that provide the remote access services to their related APOs. The AR ASE contains a single AREP that conveys service requests and responses for the ASEs to one or more remote AREPs located in remote APs.

4.5 FAL Service Procedures and Time Sequence Diagrams

This subclause contains service procedures for FAL confirmed and unconfirmed services.

4.5.1 FAL Confirmed Service Procedures

FAL confirmed services are capable of operating through queues or buffers.

The requesting user submits a confirmed service request primitive to its FAL AE and starts an associated timer to monitor the request. The appropriate FAL ASE builds the related confirmed service request APDU Body and conveys it on the specified AR using an AR Confirmed Send Service request primitive. It also creates a transaction state machine.

Upon receipt of the confirmed service request APDU Body in an AR Confirmed Send service indication primitive, the responding ASE decodes it. If a protocol error was encountered while decoding the received APDU Body, the ASE uses the AR Abort Service to abort the AR or it internally requests the AP ASE to generate a Reject-PDU (which option is taken is dependent on the AR type). If a protocol error did not occur, the ASE delivers a confirmed service indication primitive to its user.

If the responding user is able to successfully process the request, the user returns a confirmed service response (+) primitive.

If the responding user is unable to successfully process the request, the service fails and the user issues a confirmed service response (-) primitive indicating the reason.

The responding ASE builds a confirmed service response APDU Body for a confirmed service response (+) primitive or a confirmed service error APDU Body for a confirmed service response (-) primitive and conveys it on the specified AR using an AR Confirmed Send service response primitive.

Upon receipt of the returned APDU Body in an AR Confirmed Send service confirmation (+ or -) primitive the initiating ASE delivers a confirmed service confirmation primitive to its user which specifies success or failure, and the reason for failure if a failure occurred. It also cancels the corresponding transaction state machine.

However, if the AP's transaction timer expires before the corresponding response or error APDU is received, the AP may take corrective actions, which may include instructing its local ASE to cancel the transaction state machine. Such instruction to the ASE would occur through a locally defined interface.

4.5.2 Confirmed Service Time Sequence Diagram

Figure 13 illustrates the sequence of confirmed service primitives.

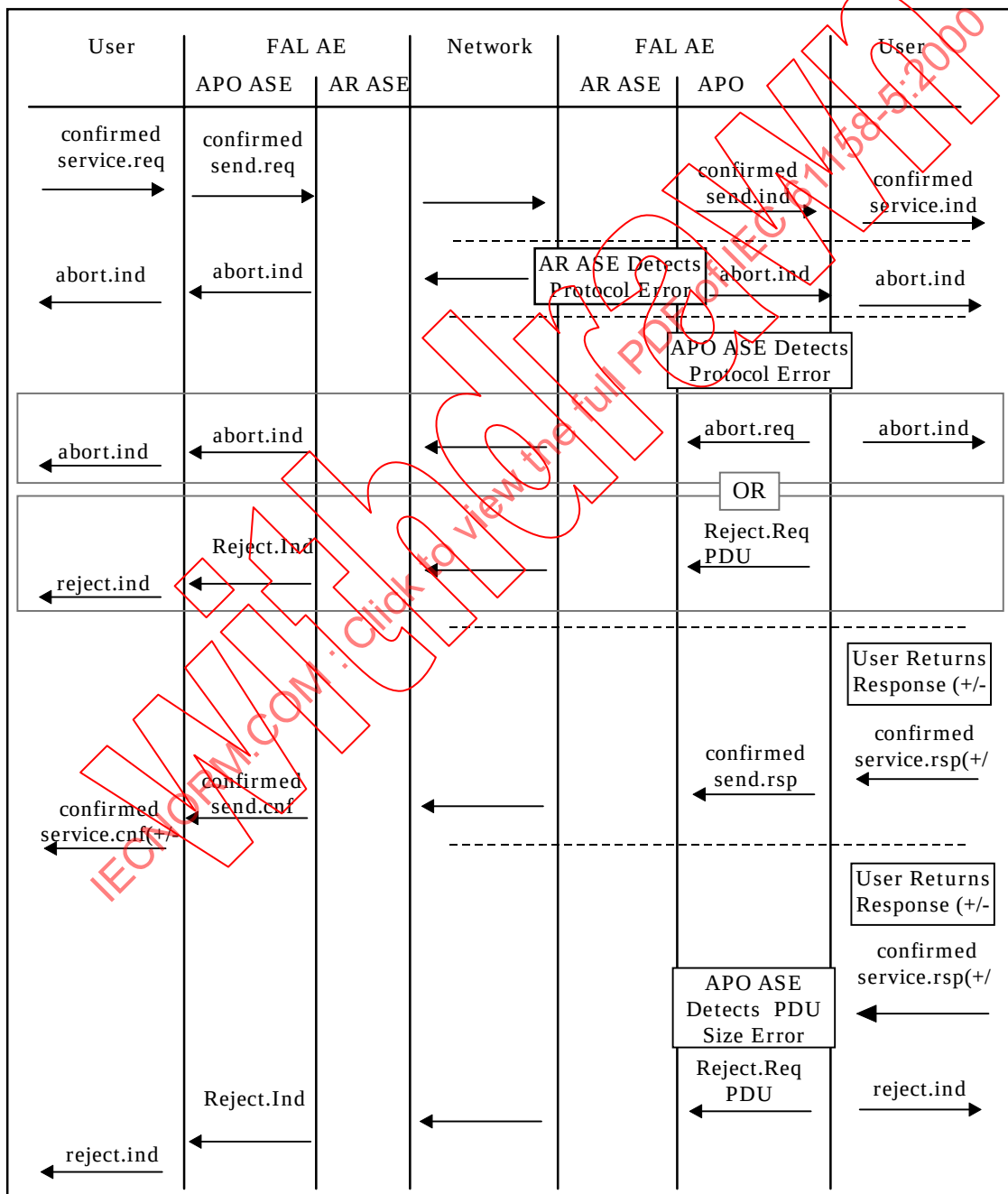


Figure 13 – Time Sequence Diagram For The Confirmed Services

4.5.3 FAL Unconfirmed Service Procedures

FAL unconfirmed services are capable of operating through queues or buffers.

The requesting user submits an unconfirmed service request primitive to its FAL AE. The appropriate FAL ASE builds the related unconfirmed service request APDU Body and conveys it on the specified AR using an AR Unconfirmed Send Service request primitive.

Upon receipt of the unconfirmed request APDU Body in an AR Unconfirmed Send Service indication primitive, the receiving ASE(s) participating in the AR delivers the appropriate unconfirmed service indication primitive to its user. Timeliness parameters are included in the indication primitive if the AR that conveyed the APDU Body supports timeliness.

4.5.4 Unconfirmed Service Time Sequence Diagram

Figure 14 illustrates the sequence of unconfirmed service primitives.

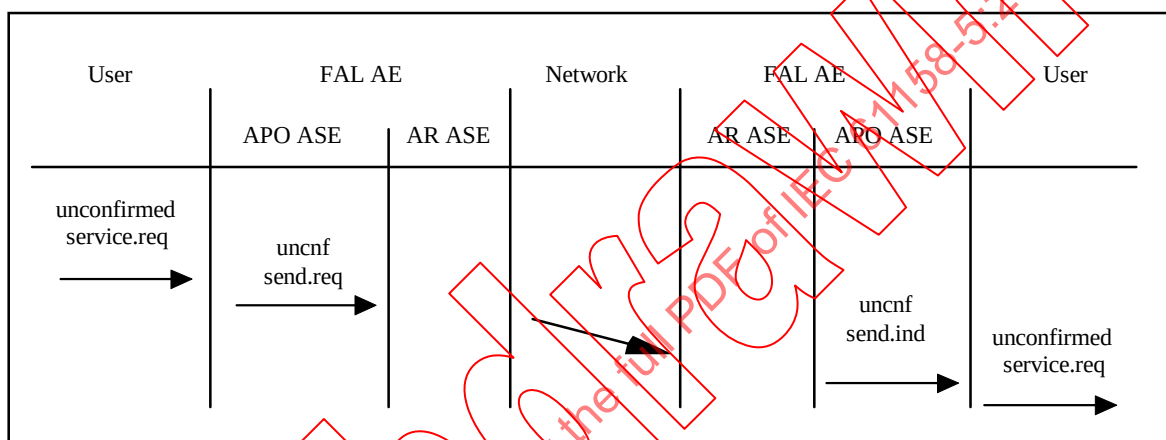


Figure 14 – Time Sequence Diagram For Unconfirmed Services

4.6 Common FAL attributes

The following attributes are defined for all FAL Class specifications of the FAL Classes that follow, except for the Data Type class. Therefore, they are defined here instead of with the other attributes for each of the services.

ATTRIBUTES:

- 1 (o) Key Attribute: Numeric Identifier
- 2 (o) Key Attribute: Name
- 3 (o) Attribute: User Description
- 4 (o) Attribute: Object Revision

Numeric Identifier

This optional key attribute specifies the numeric id of the object. It is used as a shorthand reference by the FAL protocol to identify the object. There are three possibilities for identification purposes: numeric identifier or name or both. This attribute is required for the Data Type model.

Name

This optional key attribute specifies the name of the object. There are three possibilities for identification purposes: numeric identifier or name or both.

User description

This optional attribute contains user defined descriptive information about the object.

Object Revision

This optional attribute specifies the revision level of the object. It is a structured attribute composed of major and minor revision numbers. If Object Revision is supported, it contains both a Major Revision and a Minor Revision with a value range 0 to 15 for each. The use of major/minor fields is intended to provide the following features:

Major Revision

The Major Revision field contains the major revision value for the object. A change to the major revision indicates that interoperability is affected by the change.

Minor Revision

The Minor Revision field contains the minor revision value for the object. A change to the minor revision indicates that interoperability was not affected by the change – that is users of the object will continue to be capable of interoperating with the object when its minor revision is changed, provided that the major revision remains the same.

4.7 Common FAL Service Parameters

In the specifications of the FAL services that follow, the following parameters are used frequently and consistently. Therefore, they are defined here instead of with the other parameters for each of the services.

AREP

This parameter contains sufficient information to locally identify the AREP to be used to convey the service. This parameter may use a key attribute of the AREP to identify the application relationship. When an AREP supports multiple contexts (established using the Initiate service) at the same time, the AREP parameter is extended to identify the context as well as the AREP.

Invoke ID

This parameter identifies this invocation of the service. It is used to associate service requests with responses. Therefore, no two outstanding service invocations may be identified by the same Invoke ID value.

FAL ASE/FAL Class

This parameter specifies the FAL ASE (AP, AR, Variable, Data Type, Event, Function Invocation, Load Region) and the FAL Class within the ASE (e.g. AREP, Variable List, Notifier, Action).

Numeric ID

This parameter is the numeric identifier of the object.

Error Info

This parameter provides error information for service errors. It is returned in confirmed service response(-) primitives. It is composed of the following elements.

Error Class

This parameter indicates the general class of error. Valid values are specified in the definition of Error Code parameter, below.

Error Code

This parameter identifies the specific service error. Table 1 indicates the error codes by class.

Table 1 – Error Codes by Class

Error Code	Error Class								
	AP State	AP Ref	Definition	Resource	Service	Access	Obj Def	Other	Conclude
other	X	X	X	X	X	X	X	X	X
ap-unreachable		X							
object-undefined			X						
object-attributes-inconsistent			X						
name-already-exists			X						
memory-unavailable				X					
object-state-conflict					X				
pdu-size					X				
object-constraint-conflict					X				
parameter-inconsistent					X				
illegal-parameter					X				
object-invalidated						X			
hardware-fault						X			
object-access-denied						X			
invalid-address						X			
object-attribute-inconsistent						X			
object-access-unsupported						X			
object-non-existent						X			
type-conflict						X			
named-access-unsupported						X			
access-to-element-unsupported						X			
name-length-overflow							X		
obj def-overflow							X		
obj def -write-protected							X		
AP-description-length-overflow							X		
operational-problem							X		
obj def ext overflow							X		
AP reference invalid		X							
Context unsupported		X							
Type unsupported			X						
Type inconsistent			X						
Further communication needed									X

Additional Code

This optional parameter identifies the error encountered when processing the request specific to the object being accessed. When used, the value submitted in the response primitive is delivered unchanged in the confirmation primitive.

Additional Detail

This optional parameter contains user data that accompanies the negative response. When used, the value submitted in the response primitive is delivered unchanged in the confirmation primitive.

4.8 APDU size

All data transferred in a single invocation of an FAL service is transferred in a single FAL APDU, unless the AR used to convey the service supports segmentation. When segmentation is not supported, how the AP sending the data determines whether or not the data will fit into the FAL APDU is a local matter.

It may not be possible to convey all requested data with a single invocation of this service. If the user data of a service request or response cannot be conveyed in a single APDU, then a local mechanism may be used to indicate this condition to the requester. When the user data for a response cannot be conveyed in a single service invocation, or in a set of related service invocations using the More Flag parameter, then the service fails and the user should issue a response (-) primitive indicating the reason.

IECNORM.COM: Click to view the full PDF of IEC 61158-5:2000

Withdr2Wm

5 Data Type ASE

5.1 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.

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 15. 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.

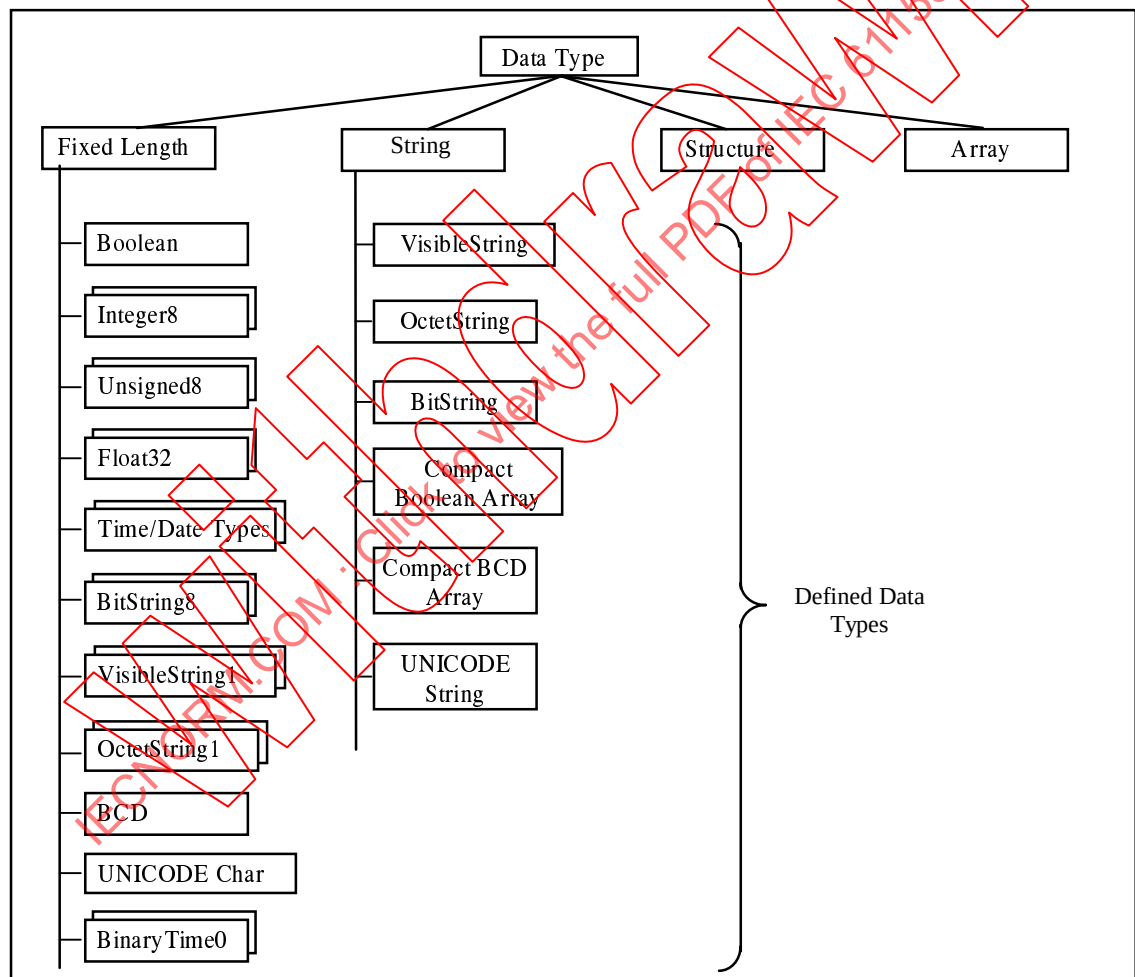


Figure 15 – Data Type Class Hierarchy

The data type definitions in Figure 15 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 15.

The basic data classes are used to define fixed length and bitstring data types. Standard types taken from ISO/IEC 8824 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.

5.1.1 Overview Of Basic Types

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

Simple types are ISO/IEC 8824 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.

5.1.1.1 Fixed Length Types

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

5.1.2 Overview of Constructed Types

Constructed data types are needed to completely convey the variety of information present on the Fieldbus. There are two kinds of constructed types defined for this standard, arrays and structures.

5.1.2.1 Strings

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

5.1.2.2 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.

5.1.2.3 Structures

A structure is made of an ordered set of heterogeneously 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.

5.1.2.4 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".

5.1.3 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.

5.1.4 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. 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

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

5.2.1.1 Attributes

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.

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

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	1
2	Data Type Name	=	Boolean
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

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

5.3.1.2 BOOL

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

5.3.1.3 Integer8

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	2
2	Data Type Name	=	Integer8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

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

5.3.1.4 SINT

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

5.3.1.5 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.

5.3.1.6 INT

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

5.3.1.7 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.

5.3.1.8 DINT

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

5.3.1.9 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.

5.3.1.10 LINT

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

5.3.1.11 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.

5.3.1.12 USINT

This IEC 61131-3 type is the same as Unsigned 8 above.

5.3.1.13 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.

5.3.1.14 UINT

This IEC 61131-3 type is the same as Unsigned 16 above.

5.3.1.15 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.

5.3.1.16 UDINT

This IEC 61131-3 type is the same as Unsigned 32 above.

5.3.1.17 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.

5.3.1.18 ULINT

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

5.3.1.19 Float32

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	8
2	Data Type Name	=	Float32
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type has a length of four octets. The format for float32 is that defined by ANSI/IEEE 754 as single precision.

5.3.1.20 Floating Point

This data type is the same as Float32 above.

5.3.1.21 REAL

This IEC 61131-3 type is the same as Float 32 above.

5.3.1.22 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 ANSI/IEEE 754 as double precision.

5.3.1.23 LREAL

This IEC 61131-3 type is the same as Float 64 above.

5.3.1.24 BinaryDate**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	11
2	Data Type Name	=	BinaryDate
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. 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.

5.3.1.25 BinaryDate2000**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	51
2	Data Type Name	=	BinaryDate2000
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

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. 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 Unsigned16 data type and gives the year.

5.3.1.26 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.

5.3.1.27 DATE

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	<tbid>
2	Data Type Name	=	DATE
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This IEC 61131-3 type is a binary number. The most significant bit of the most significant 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. It expresses the date as a number of days, starting from D#1972-01-01 (January 1st, 1972), the start of the Coordinated Universal Time (UTC) era, until D#2151-06-06 (June 6th, 2151), i.e. a total range of 65536 days.

5.3.1.28 TimeOfDay

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	12
2	Data Type Name	=	TimeOfDay
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

This data type is composed of two elements of unsigned values and expresses the time of day and the date. The first element is an Unsigned32 data type and gives the time after the midnight in milliseconds. The second element is an Unsigned16 data type and gives the date.

5.3.1.29 TimeOfDay with date indication

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

5.3.1.30 TimeOfDay without date indication

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	52
2	Data Type Name	=	TimeOfDay without date indication
4	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

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

5.3.1.31 TIME_OF_DAY

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

5.3.1.32 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.

5.3.1.33 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 days.

5.3.1.34 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.

5.3.1.35 TimeValue**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	21
2	Data Type Name	=	Time Value
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This simple type expresses the time or time difference in a two's complement binary number with a length of eight octets. The unit of time is 1/32 millisecond.

5.3.1.36 UniversalTime

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	16
2	Data Type Name	=	UniversalTime
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	12

This simple type is composed of twelve elements of type VisibleString. (YYMMDDHHMMSS). It is the same as that defined in ISO/IEC 8824, except that the local time differential is not supported.

5.3.1.37 FieldbusTime

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	17
2	Data Type Name	=	FieldbusTime
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	7

This data type is defined in IEC 61158-4 of this standard as DL-Time.

5.3.1.38 BitString8

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	22
2	Data Type Name	=	Bitstring8
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	1

This type contains 1 element of type BitString.

5.3.1.39 BitString16

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	23
2	Data Type Name	=	Bitstring16
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	2

5.3.1.40 BitString32

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	24
2	Data Type Name	=	Bitstring32
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	4

5.3.1.41 BitString64**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	57
2	Data Type Name	=	Bitstring64
3	Format	=	FIXED LENGTH
5.1	Octet Length	=	8

5.3.1.42 BYTE

This IEC 61131-3 type is the same as Bitstring 8 above.

5.3.1.43 WORD

This IEC 61131-3 type is the same as Bitstring 16 above.

5.3.1.44 DWORD

This IEC 61131-3 type is the same as Bitstring 32 above.

5.3.1.45 LWORD

This IEC 61131-3 type is the same as Bitstring 64 above.

5.3.1.46 VisibleString1**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	25
2	Data Type Name	=	VisibleString1
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This type is defined as a single character in the ISO VisibleString type.

5.3.1.47 VisibleString2**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	26
2	Data Type Name	=	VisibleString2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This type contains two elements of type VisibleString.

5.3.1.48 VisibleString4

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	27
2	Data Type Name	=	VisibleString4
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type contains four elements of type VisibleString.

5.3.1.49 VisibleString8

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	28
2	Data Type Name	=	VisibleString8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This type contains eight elements of type VisibleString.

5.3.1.50 VisibleString16

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	29
2	Data Type Name	=	VisibleString16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	16

This type contains 16 elements of type VisibleString.

5.3.1.51 OctetString1

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	30
2	Data Type Name	=	OctetString1
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This type has a length of one octet.

5.3.1.52 OctetString2

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	31
2	Data Type Name	=	OctetString2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This type has a length of two octets.

5.3.1.53 OctetString4**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	32
2	Data Type Name	=	OctetString4
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type has a length of four octets.

5.3.1.54 OctetString8**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	33
2	Data Type Name	=	OctetString8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This octet string type has a length of eight octets.

5.3.1.55 OctetString16**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	34
2	Data Type Name	=	OctetString16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	16

This type has a length of 16 octets.

5.3.1.56 BCD**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	35
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. In this type, the least significant four bits are used to express a BCD value which is between zero and nine inclusive. The most significant four bits are unused.

5.3.1.57 UNICODE Char

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	36
2	Data Type Name	=	UnicodeChar
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This type is defined as a single character in the UNICODE string type.

5.3.1.58 BinaryTime0

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	40
2	Data Type Name	=	BinaryTime0
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 type has a length of two octets. The unit of time for this type is 10 µs.

5.3.1.59 BinaryTime1

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	41
2	Data Type Name	=	BinaryTime1
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 type has a length of two octets. The unit of time for this type is 100 µs.

5.3.1.60 BinaryTime2

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	42
2	Data Type Name	=	BinaryTime2
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 type has a length of two octets. The unit of time for this type is 1 ms.

5.3.1.61 BinaryTime3**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	43
2	Data Type Name	=	BinaryTime3
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 type has a length of two octets. The unit of time for this type is 10 ms.

5.3.1.62 BinaryTime4**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	44
2	Data Type Name	=	BinaryTime4
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 type has a length of four octets. The unit of time for this type is 10 μ s.

5.3.1.63 BinaryTime5**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	45
2	Data Type Name	=	BinaryTime5
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 type has a length of four octets. The unit of time for this type is 100 μ s.

5.3.1.64 BinaryTime6**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	46
2	Data Type Name	=	BinaryTime6
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 type has a length of four octets. The unit of time for this type is 1 ms.

5.3.1.65 BinaryTime7

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	47
2	Data Type Name	=	BinaryTime7
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 type has a length of six octets. The unit of time for this type is 1 ms.

5.3.1.66 BinaryTime8

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	48
2	Data Type Name	=	BinaryTime8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

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 six octets. The unit of time for this type is 10 µs.

5.3.1.67 BinaryTime9

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	49
2	Data Type Name	=	BinaryTime9
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

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 binary time type has a length of six octets. The unit of time for this type is 100 µs.

5.3.1.68 TIME

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	<tbd>
2	Data Type Name	=	TIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

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

5.3.1.69 ITIME**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	<tbd>
2	Data Type Name	=	ITIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

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

5.3.1.70 FTIME**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	<tbd>
2	Data Type Name	=	FTIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

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

5.3.1.71 LTIME**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	<tbd>
2	Data Type Name	=	LTIME
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

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

5.3.2 String Types**5.3.2.1 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 string type.

5.3.2.2 OctetString

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	10
2	Data Type Name	=	OctetString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

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

5.3.2.3 BitString

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	14
2	Data Type Name	=	Bitstring
3	Format	=	STRING
5.1	Octet Length	=	1

This string type is defined as a series of eight bits, numbered from 0 to 7.

5.3.2.4 CompactBooleanArray

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	37
2	Data Type Name	=	CompactBooleanArray
3	Format	=	STRING
6.1	Octet Length	=	1

In this type, each bit value of 0 (zero) represents Boolean value FALSE and each bit value of 1 represents TRUE.

5.3.2.5 CompactBCDArray

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	38
2	Data Type Name	=	CompactBCDArray
3	Format	=	STRING
4.1	Octet Length	=	1

This type is used to pack an ordered series of BCD values, two per octet, into an ordered series of octets. The first octet contains the most significant BCD value in its most significant quartet. If the number of BCD values is odd, the least significant quartet of the final octet is set to the reserved value "1111".

5.3.2.6 UNICODEString**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	39
2	Data Type Name	=	UnicodeString
3	Format	=	STRING
4.1	Octet Length	=	2

This type is defined as the UNICODE string type.

5.3.3 Structure Types**5.3.3.1 DATE_AND_TIME****CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	<tbd>
2	Data Type Name	=	DATE_AND_TIME
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Time_Of_Day_Element
5.2.2	Field Data Type	=	TIME_OF_DAY
5.3.1	Field Name	=	Date_Element
5.3.2	Field Data Type	=	DATE

This IEC 61131-3 type extension is a structure which expresses both the date (as a number of days starting from January 1st, 1972 until June 6th, 2151), and the time of day as a number of ms starting from midnight.

5.3.3.2 STRING**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	<tbd>
2	Data Type Name	=	STRING
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data Type	=	UINT
5.3.1	Field Name	=	Stringcontents_Element
5.3.2	Field Data Type	=	OctetString

This IEC 61131-3 type is composed of two elements. Charcount_Element gives the current number of characters in the Stringcontents_Element (one USINT per character).

5.3.3.3 STRING2

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	<tbid>
2	Data Type Name	=	STRING2
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data Type	=	UINT
5.3.1	Field Name	=	String2contents_Element
5.3.2	Field Data Type	=	OctetString

This IEC 61131-3 data type extension is composed of two elements. Charcount_Element gives the current number of characters in the String2contents_Element (one UINT per character).

5.3.3.4 STRINGN

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	<tbid>
2	Data Type Name	=	STRINGN
3	Format	=	STRUCTURE
5.1	Number of Fields	=	3
5.2.1	Field Name	=	Charsize_Element
5.2.2	Field Data Type	=	UINT
5.3.1	Field Name	=	Charcount_Element
5.3.2	Field Data Type	=	UINT
5.4.1	Field Name	=	StringNcontents_Element
5.4.2	Field Data Type	=	OctetString

This IEC 61131-3 type extension is composed of three elements. Charsize_Element gives the size of a character in StringNcontents_Element (N = number of USINT). Charcount_Element gives the current number of characters in the StringNcontents_Element (N USINT per character).

5.3.3.5 SHORT_STRING

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	??
2	Data Type Name	=	SHORT_STRING
3	Format	=	STRUCTURE
5.1	Number of Fields	=	2
5.2.1	Field Name	=	Charcount_Element
5.2.2	Field Data Type	=	USINT
5.3.1	Field Name	=	Stringcontents_Element
5.3.2	Field Data Type	=	OctetString

This IEC 61131-3 type extension is composed of two elements. Charcount_Element gives the current number of characters in the Stringcontents_Element (one USINT per character).

5.4 Data type ASE Service Specification

There are no operational services defined for the type object.

5.5 Summary of data types

This clause contains a summary of the defined data types. The Class ID values have been assigned to be compatible with existing standards. Table 2 is sorted by class id.

Table 2 – Data Type Summary

Class Id	Data Type	Class Id	Data Type
1	Boolean	30	OctetString1
2	Integer8	31	OctetString2
3	Integer16	32	OctetString4
4	Integer32	33	OctetString8
5	Unsigned8	34	OctetString16
6	Unsigned16	35	BCD
7	Unsigned32	36	UNICODE Char
8	Float32	37	CompactBooleanArray
8	FloatingPoint	38	CompactBCDArray
9	VisibleString	39	UNICODEString
10	OctetString	40	BinaryTime0
11	BinaryDate	41	BinaryTime1
12	TimeOfDay	42	BinaryTime2
12	TimeOfDay with date indication	43	BinaryTime3
13	TimeDifference	44	BinaryTime4
14	BitString	45	BinaryTime5
15	Float64	46	BinaryTime6
16	UniversalTime	47	BinaryTime7
17	FieldbusTime	48	BinaryTime8
21	TimeValue	49	BinaryTime9
22	BitString8	50	Date
23	BitString16	51	BinaryDate2000
24	BitString32	52	TimeOfDay without date indication
25	VisibleString1	53	TimeDifference with date indication
26	VisibleString2	54	TimeDifference without date indication
27	VisibleString4	55	Integer64
28	VisibleString8	56	Unsigned64
29	VisibleString16	57	Bitstring64

6 Types of Communication Models

This clause defines different types of communications models. Each type of model defines a specific set of ASEs and ARs that it uses for communications. All types use the data types defined in Clause 5. All are structured according to concepts defined in Clause 4. However, each type may have define own additional concepts. The types are:

Type 1	Existing IEC
Type 2	Control-Net
Type 3	Profibus
Type 4	P-NET
Type 5	FF HSE
Type 6	Swiftnet
Type 7	WorldFip
Type 8	INTERBUS

7 Communication Model Type 1 Specification

7.1 Type 1 Concepts

There are no additional concepts for the Type 1 Communications Model

7.2 Type 1 ASEs

7.2.1 Object Management ASE

7.2.1.1 Overview

The FAL ASE Models each specify one or more related FAL APO classes as a collection of attributes and services. The FAL services defined for a class are used to manipulate, control, or access APOs of the class. The services which are supported by a specific APO are specified by the ListOfManagementServices attribute of the APO.

To dynamically create or delete the network view of an APO, or to access its attribute values, this FAL ASE defines common FAL APO management services. Each APO Class definition specifies which of these services may be included in the definitions of APOs of the class. These services operate using the client/server model.

7.2.1.2 FAL Management Model Specification

The management services specified by the FAL Management Model operate on FAL APOs and their attributes. FAL APO classes are defined within the FAL ASEs that are specified in the remaining clauses of this standard. Each FAL APO Class defined is described by a set of attributes and a set of services.

7.2.1.3 FAL Management Model Services

Management services are defined for managing APs and APOs. For simplicity APs and APOs are referred to as objects in the remainder of this clause.

Create and Delete services are specified for a class if the network view of an object of the class can be dynamically created or deleted. Get Attributes and Set Attributes services are specified to indicate when its attributes can be read or written. The attributes of an APO which can be set are specified as part of the attribute definition.

Because unexpected changes to the object definitions within an AP can affect operation of the AP, the Begin Set Attributes and End Set Attributes services are defined. The Begin Set Attributes service is used to request the remote AP for permission to change its object definitions. The End Set Attributes service is used to indicate to the remote AP that changes to its object definitions are complete and that it can resume normal operation.

7.2.1.3.1 Create Service

This confirmed service is used to make an object visible and accessible across the network. As part of the service, initial values for the object's attributes may be specified. If all attribute values to be initialized cannot be conveyed in a single APDU, then the Set Attributes service can be used once the object has been created for attributes that may be updated using the Set Attributes service. The scope of visibility for created objects is Application Process specific.

The List of Supported Attributes parameter is used to indicate to the requester the attributes which were actually created for the object. If initial values were supplied for attributes that were not created, the List of Supported Attributes can be used to determine the ones which were not created and initialized according to the request. This capability represents a form of negotiation when creating an object.

NOTE Whether or not an AP actually creates the object is a local matter. The intent of the service is to request that it be prepared for remote access.

7.2.1.3.1.1 Service Parameters

The service parameters for this service are shown in Table 3.

Table 3 – Create Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
FALSE / FAL Class	M	M(=)		
List of Attribute IDs and Initial Values	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Numeric ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

List of Attribute IDs and Initial Values

This parameter identifies and specifies the initial value of one or more of the object's attributes to be set during the creation process. At least one of the object's key attributes must be present. Values for mandatory and optional attributes may be supplied. The AP that creates the object determines the complement of attributes supported for the object.

NOTE Initial values are only required for mandatory attributes if the AP creating the object does not have the ability to supply an initial value on its own. How it acquires these values on its own is beyond the scope of this standard.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.1.3.1.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.1.3.2 Delete Service

This confirmed service is used to make an object that is currently visible and accessible not visible and not accessible across the network.

NOTE Whether or not the real object is actually deleted or simply removed from network accessibility is a local matter.

7.2.1.3.2.1 Service Parameters

The service parameters for this service are shown in Table 4.

Table 4 – Delete Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
FAL ASE / FAL Class	M	M(=)		
Key Attribute	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

FAL ASE/FAL Class

This parameter specifies the FAL ASE (AP, AR, Variable, Data Type, Event, Function Invocation, Load Region) and the FAL Class within the ASE (e.g. AREP, Variable List, Notifier, Action).

Key Attribute

This parameter identifies the object by one of its key attributes.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.1.3.2.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

Once deleted, an object is no longer accessible using the services of the FAL.

7.2.1.3.3 Get Attributes Service

This confirmed service is used to read the current values of a list of attributes for one or more objects of one or more classes defined for the AP. It operates in a "best effort" fashion, such that the service succeeds if the value for at least one requested attribute for one requested object is returned.

7.2.1.3.3.1 Service Parameters

The service parameters for this service are shown in Table 5.

Table 5 – Get Attributes Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
List of Objects and their Attributes	S	S(=)		
Key Attribute	M	M(=)		
List of Requested Attributes	M	M(=)		
Data Type Requested	C	C(=)		
Range of Objects and their Attributes	S	S(=)		
Starting Numeric ID	M	M(=)		
List of Requested Attributes	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
More			M	M(=)
List of Responses			M	M(=)
List of Supported Attributes			C	C(=)
Error Status			S	S(=)
List of Attribute Values			S	S(=)
Data Type Description			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

List of Objects and their Attributes

This selection type parameter provides a list that identifies each object, its class, and the attributes that are being requested. Three values for identifying attributes are possible, all attributes, mandatory attributes only, or a list of individual attributes. It is present if the user is requesting attributes for a list of objects.

Key Attribute

This parameter identifies the object for which attributes are being requested using one of the object's key attributes.

List of Requested Attributes

This parameter identifies one or more of the attributes for which values are being requested.

Range of Objects and their Attributes

This selection type parameter provides a range of numeric identifiers for objects of a single class, and optionally for a series of objects of different classes, and the attributes that are being requested. Three values for identifying attributes are possible, all attributes, mandatory attributes only, or a list of individual attributes. It is present if the user is requesting attributes for objects within a given range of numeric ids.

Starting Numeric ID

This parameter specifies the first in a range of numeric ids. The attributes being requested are for all objects beginning with and after this numeric id. The value ALL may be used to indicate that attributes for all objects for the specified class are to be returned.

NOTE Multiple requests may have to be issued with successively higher numeric ids when not all the desired object definitions can be returned in a single response.

List of Requested Attributes

This parameter identifies one or more of the attributes for which values are being requested.

Data type Requested

This conditional parameter is present only when the attributes of a variable object are being requested. It indicates whether the data type description is to be returned with requested variable attributes.

Result(+)

This selection type parameter indicates that the service request succeeded.

More

This Boolean parameter indicates, when FALSE, that responses for all requested attributes are being returned. If the value is TRUE, then only responses for a proper subset of the attributes are being returned. This situation will occur only when not all values will fit into a single response APDU. When it does occur, it is necessary for the user to submit an additional request for the missing attributes. Partial values for attributes are never returned.

List of Responses

This parameter contains a status indicator or a list of attribute values for each object specified in the request. The responses in the list are returned in the same order that the objects were specified in the request. The object associated with each response is only explicitly identified if the value of one of its key attributes was requested. If all responses could not be returned in a single APDU, the More parameter is set.

List of Supported Attributes

This parameter identifies the attributes supported by the responder for the specified object. It is present if the specified object exists.

Error Status

This parameter is returned if the get failed for all of the attributes of an object. It indicates the most probable reason why the operation failed using the error class and error code defined for the Get Attributes service.

List of Attribute Values

This parameter is returned if the Get succeeded for any one or the requested attributes. This parameter contains a list of attribute values, each individually identified and complete, for the object requested.

Data Type Description

This conditional parameter is present only when data type description for a variable object has been requested. It contains the complete data type description for the variable.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.1.3.3.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Get Attributes Service is a "best effort". Best effort means that the service succeeds if at least one value is returned. If the user is able to get one or more of the specified attribute values, and if they can be conveyed in a single APDU, they are returned in a Get Attributes Service response (+) primitive.

If they cannot be conveyed in a single APDU, the user returns those that can and sets the more flag in the Get Attributes Service response (+) primitive. The requester is then responsible for submitting additional requests identifying the remaining attributes to be retrieved.

If the user is unable to get the value for at least one attribute for one object in the list, the service fails and the user issues a Get Attributes Service response (-) primitive indicating the reason.

7.2.1.3.4 Set Attributes Service

This confirmed service is used to update the current value of a list of attributes for one or more objects. This service operates in an all or nothing fashion — if all the requested attributes for all the objects cannot be set, the service fails.

NOTE This service may be used to load entire object definitions by indicating that all mandatory attributes are to be updated.

7.2.1.3.4.1 Service Parameters

The service parameters for this service are shown in Table 6.

Table 6 – Set Attributes Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
List of Object Definitions	M	M(=)		
FAL ASE / FAL Class	M	M(=)		
Key Attribute	M	M(=)		
List of Attribute Updates	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)
List of Supported Attributes			C	C(=)

Argument

The argument contains the parameters of the service request.

List of Object Updates

This parameter contains the list of objects and their attributes that are to be updated.

FAL ASE/FAL Class

This parameter specifies the FAL ASE (AP, AR, Variable, Data Type, Event, Function Invocation, Load Region) and the FAL Class within the ASE (e.g. AREP, Variable List, Notifier, Action).

Key Attribute

This parameter identifies the object by one of its key attributes.

List of Attribute Updates

This parameter identifies one or more of the attributes that are to be updated and their values.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

List of Supported Attributes

This conditional parameter identifies the attributes that are supported by the object. It is present when a request was received to update unsupported attributes.

7.2.1.3.4.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Set Attributes Service is an "all or nothing" service. All or nothing means that the service succeeds only if all the requested updates succeed.

7.2.1.3.5 Begin Set Attributes Service

The Begin Set Attributes prepares the AP for updates to its object definitions, such that the updates are free of interaction or not free of interaction. The AP returns a successful response to indicate to the requester that it may begin updating object definitions.

7.2.1.3.5.1 Service Parameters

The service parameters for this service are shown in Table 7.

Table 7 – Begin Set Attributes

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Consequence	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Consequence

This parameter states whether the intended modifications are free of interaction for the other communication partners, not free of interaction with updates to individual object definitions, or not free of interaction with completely new load of all object definitions.

- loading free of interaction
- reload, not free of interaction
- new load, not free of interaction

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.1.3.5.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.1.3.6 End Set Attributes Service

The End Set Attributes service terminates the process of updating object definitions. It is used to indicate to the AP that all object definition updates have been completed.

7.2.1.3.6.1 Service Parameters

The service parameters for this service are shown in Table 8.

Table 8 – End Set Attributes

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)
Numeric ID			C	C(=)

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Numeric ID

This parameter gives the Numeric ID of the object, for which the generation was not successful.

7.2.1.3.6.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.2 Application Process ASE

7.2.2.1 Overview

This standard models a Fieldbus Application Process (AP). Fieldbus Application Processes represent the information and processing resources of a system that can be accessed through FAL services.

The Application Service Element in the FAL that provides these services is called an Application Process ASE. In the AP ASE, the AP is modeled and accessed as an APO with a standardized and predefined identifier.

7.2.2.2 AP Class Specification

7.2.2.2.1 AP Formal Model

The AP class specifies the attributes and services defined for application processes. Its parent class "top" indicates the top of the FAL class tree. The numeric identifier for the AP Object is always 0. The use of this reserved value permits management services to be used for AP objects without knowing their names.

FAL ASE: **AP ASE**

CLASS: AP

CLASS ID: 16

PARENT CLASS: TOP

IDENTIFY, GET STATUS, and STATUS NOTIFICATION SERVICE ATTRIBUTES:

- 1 (m) Attribute: List Of AP Service Attributes
- 1.1 (m) Attribute: Manufacturer Identifier
- 1.1.1 (m) Attribute: Vendor Name
- 1.1.2 (m) Attribute: Model Identifier
- 1.1.3 (m) Attribute: Vendor Revision
- 1.1.4 (o) Attribute: Serial Number
- 1.2 (o) Attribute: AP Descriptor Reference
- 1.3 (m) Attribute: Network Access State
- 1.3.1 (m) Attribute: Service Access Status
- 1.3.2 (m) Attribute: Operational Status

SYSTEM MANAGEMENT ATTRIBUTES:

2	(m)	Attribute:	List Of System Management Attributes
2.1	(m)	Attribute:	List Of AR Endpoint Info
2.1.1	(m)	Attribute:	Numeric Id
2.1.2	(m)	Attribute:	Configured Max FAL PDU Size Sending
2.1.3	(m)	Attribute:	Configured Max FAL PDU Size Receiving
2.1.4	(m)	Attribute:	Configured Max Outstanding Services Requesting
2.1.5	(m)	Attribute:	Configured Max Outstanding Services Responding
2.1.6	(m)	Attribute:	List of Supported Services
2.1.7	(o)	Attribute:	Configured Max Data Structure Nesting Level
2.2	(m)	Attribute:	List Of In Use AR Endpoint Info
2.2.1	(m)	Attribute:	Context State
2.2.2	(m)	Attribute:	Negotiated Max FAL PDU Size Sending
2.2.3	(m)	Attribute:	Negotiated Max FAL PDU Size Receiving
2.2.4	(m)	Attribute:	Negotiated Max Outstanding Services Requesting
2.2.5	(m)	Attribute:	Negotiated Max Outstanding Services Responding
2.2.6	(m)	Attribute:	Outstanding Services Requesting Counter
2.2.7	(m)	Attribute:	Outstanding Services Responding Counter
2.2.8	(o)	Attribute:	Negotiated Max Data Structure Nesting Level
2.2.9	(o)	Attribute:	Negotiated List of Supported Services

OBJECT MANAGEMENT ATTRIBUTES:

3	(m)	Attribute:	List Of Managed AP Attributes
3.1	(m)	Attribute:	List Of Supported APO Classes and Objects
3.1.1	(m)	Attribute:	List Header
3.1.1.1	(o)	Attribute:	Numeric Identifier = 0
3.1.1.2	(m)	Attribute:	ROM / RAM Flag (TRUE, FALSE)
3.1.1.3	(m)	Attribute:	Max Name Length
3.1.1.4	(m)	Attribute:	Access Protection Supported (TRUE, FALSE)
3.1.1.5	(m)	Attribute:	Version Of List
3.1.1.6	(m)	Attribute:	List Local Reference
3.1.2	(c)	Constraint	Data Type Supported
3.1.2.1	(o)	Attribute:	Numeric Identifier = 1
3.1.2.2	(m)	Attribute:	Number Of Data Type Objects
3.1.2.3	(m)	Attribute:	Local Reference
3.1.3	(c)	Constraint	Variable Load Region Event Supported
3.1.3.1	(o)	Attribute:	Static Object Numeric Identifier
3.1.3.2	(m)	Attribute:	Number Of Static Objects (Variables + Load Region + Events)
3.1.3.3	(m)	Attribute:	Static Object Local Reference
3.1.4	(c)	Constraint	Variable List Supported
3.1.4.1	(o)	Attribute:	Numeric Identifier
3.1.4.2	(m)	Attribute:	Number Of Variable List Objects
3.1.4.3	(m)	Attribute:	Variable List Local Reference
3.1.5	(c)	Constraint	Function Invocation Supported
3.1.5.1	(o)	Attribute:	Function Invocation Numeric Identifier
3.1.5.2	(m)	Attribute:	Number Of Function Invocation Objects
3.1.5.3	(m)	Attribute:	Function Invocation Local Reference
3.1.6	(c)	Constraint	Dynamic Load Region
3.1.6.1	(o)	Attribute:	Load Region Numeric Identifier
3.1.6.2	(m)	Attribute:	Number Of Load Region Objects
3.1.6.3	(m)	Attribute:	Load Region Local Reference
3.2	(o)	Attribute:	List Of DLSAP Addresses
3.3	(o)	Attribute:	List Of Event Summary Selection Criteria

SUPPORTED ATTRIBUTES:

5	(o)	Attribute:	List Of Supported Attributes

SERVICES:

1	(o)	Ops Service:	Subscribe
2	(o)	Ops Service:	Identify
3	(o)	Ops Service:	Get Status
4	(o)	Ops Service:	Status Notification
5	(o)	Ops Service:	Initiate
6	(o)	Ops Service:	Terminate
7	(o)	Ops Service:	Conclude
8	(o)	Ops Service:	Reject

7.2.2.2.1.1 Identify, Get Status, and Status Notification Service Attributes

List Of AP Service Attributes

The following attributes are accessible using the Identify, Get Status, and Status Notification Services. They are not otherwise accessible.

Manufacturer Identifier

The Manufacturer Identifier is composed of the Vendor Name, Model Identifier, Vendor Revision, and optionally, the Serial Number of the AP. These attributes may be read using the Identify service.

Vendor Name

This attribute contains the name of the manufacturer of the AP.

Model Identifier

This attribute specifies the model of the AP.

Vendor Revision

This attribute specifies the revision level of the AP as defined by the manufacturer.

Serial Number

This optional attribute specifies the serial number of the AP.

NOTE The serial number may be used as a qualifier to uniquely identify an AP.

AP Descriptor Reference

This attribute is a reference to a generic description of the AP. The descriptor is an identifier for the characteristics of the AP independent of its use. Two APs, therefore, may share the same generic description. The value, and the permissible set of values for this attribute is user defined. This attribute is used in the Initiate service.

Network Access State

The Network Access State attribute defines the ability of the AP to communicate. Two components are specified in this standard, Service Access Status and Operational Status. These attributes may be read using the Get Status service. The AP may choose to report them without being requested using the Status Notification service.

Service Access Status

This attribute contains information about the state of the communication capabilities of the AP.

READY FOR COMMUNICATION All services may be used normally.

LIMITED NUMBER OF SERVICES Limited number of services may be supported in some cases (e.g. for small devices).

LOADING-NON-INTERACTING Object Definitions are in the process of being loaded locally. Therefore, the Begin Set Attributes service may not be used until the local load is complete and the state has been changed.

LOADING-INTERACTING Object Definitions are in the process of being loaded over the network (using the Set Attributes service). On the AR used for the loading, only the following services should be used:

Abort	Get Attributes
Status	Set Attributes
Identify	End Set Attributes

OperationalStatus

This attribute gives the operational state of the real AP, as follows.

OPERATIONAL

PARTIALLY OPERATIONAL

NOT OPERATIONAL

NEEDS MAINTENANCE

7.2.2.2.1.2 System Management Attributes

List Of System Management Attributes

The following attributes are accessible through System Management. They are used by all ASEs of the AP to enforce confirmed service flow control (see the state machines of IEC 61158-6). They are not otherwise accessible.

List of AR Endpoint Info

This attribute specifies the numeric identifiers and other configuration information (the AP Context attributes) of AREPs associated with the AP.

Numeric ID

This attribute specifies the numeric id for the AREP.

Configured Max FAL PDU Size Sending

For non-segmenting ARs, this attribute specifies the configured maximum number of octets that can be sent from this AREP. Its value is equal to the maximum Data Link Service Data Unit size minus one that may be sent from this AREP. For segmenting ARs, it specifies the maximum number of 256-byte segments that can be sent on this AREP.

Configured Max FAL PDU Size Receiving

For non-segmenting ARs, this attribute specifies the configured maximum number of octets that can be received on this AREP. Its value is equal to the maximum Data Link Service Data Unit size minus one that may be received on this AREP. For segmenting ARs, it specifies the maximum number of 256-byte segments that can be received on this AREP.

Configured Max Outstanding Services Requesting (ConfigMaxOsReq)

This attribute specifies the maximum number of outstanding confirmed services allowed in the client role of this AREP. Its value is configured before establishment of the AR. This attribute is used on client-server and peer-to-peer ARs (i.e. QUB) and corresponds to the number of sending transaction state machines.

Configured Max Outstanding Services Responding (ConfigMaxOsRsp)

This attribute specifies the maximum number of outstanding confirmed services allowed for this AREP as a server. Its value is configured before establishment of the AR. This attribute is used on client-server and peer-to-peer ARs (i.e., QUB) and corresponds to the number of receiving transaction state machines.

List of Supported Services

This attribute specifies the services that the FAL supports as a requester and as a responder. Support as a requester indicates that the FAL provides the ability for the AP to initiate confirmed and unconfirmed request primitives and to receive corresponding confirmed service confirmation primitives from the FAL. Support as a responder indicates that the FAL provides the ability for the AP to deliver confirmed and unconfirmed service indication primitives to the AP and receive confirmed service response primitives from the AP.

Configured Max Data Structure Nesting Level (ConfigMaxDSNestingLevel)

This optional attribute specifies the maximum number of nesting levels configured for the AR. It is present only when more than one nesting level is supported. That is, the default value for this parameter is one.

List of In-Use AR Endpoint Info

This attribute specifies dynamic information for the AREPs associated with the AP which are participating in an established AR-I or which are involved in the AR establishment process.

Context State

This attribute specifies the state of the AREP as seen by the AP. The values are:

CLOSED

OPEN

OPENING-REQUESTING

OPENING-RESPONDING

Negotiated Max FAL PDU size sending (NegMaxPduSnd)

For non-segmenting ARs, this attribute specifies the negotiated maximum number of octets that can be sent from this AREP. Its value is equal to the maximum Data Link Service Data Unit size minus one that may be sent from this AREP. For segmenting ARs, it specifies the negotiated maximum number of 256-byte segments that can be sent on this AREP.

Negotiated Max FAL PDU size receiving (NegMaxPduRcv)

For non-segmenting ARs, this attribute specifies the negotiated maximum number of octets that can be received on this AREP. Its value is equal to the maximum Data Link Service Data Unit size minus one that may be received on this AREP. For segmenting ARs, it specifies the maximum number of 256-byte segments that can be received on this AREP.

Negotiated Max outstanding services requesting (NegMaxOsReq)

This attribute specifies the maximum number of outstanding confirmed services allowed for this AREP as a client. Its value is negotiated during establishment of the AR to represent the lower value of the local Configured Max Outstanding Services Requesting attribute and the remote Configured Max Outstanding Services Responding attribute.

Negotiated Max outstanding services responding (NegMaxOsRsp)

This attribute specifies the maximum number of outstanding confirmed services allowed for this AREP as a server. Its value is negotiated during establishment of the AR to represent the lower value of the local Configured Max Outstanding Services Responding attribute and the remote Configured Max Outstanding Services Requesting attribute.

Outstanding services requesting counter (OsReqCtr)

This attribute contains the number of currently outstanding confirmed services on this AR as a requester.

Outstanding services responding counter (OsRspCtr)

This attribute contains the number of currently outstanding confirmed services on this AR as a responder.

Negotiated max data structure nesting level (NegConfigMaxDSNestingLevel)

This optional attribute specifies the maximum number of nesting levels negotiated for the AR. It is present only when more than one nesting level is supported. Its value is negotiated as the lower value configured for the two communicating APs.

Negotiated list of supported services

This optional attribute specifies the negotiated services that the FAL supports as a requester and as a responder on this AR. Support as a requester indicates that the FAL provides the ability for the AP to initiate confirmed and unconfirmed request primitives and to receive corresponding confirmed service confirmation primitives from the FAL. Support as a responder indicates that the FAL provides the ability for the AP to deliver confirmed and unconfirmed service indication primitives to the AP and receive confirmed service response primitives from the AP.

7.2.2.2.1.3 Object Management Attributes

List Of Managed AP Attributes

This attribute contains the AP attributes that are accessible using the services of the Management ASE.

List of Supported APO Classes and Objects

This attribute contains the Object Definitions maintained by the AP.

List Header

This attribute describes the characteristics of the Object Definitions maintained by the AP.

Numeric ID

This attribute is the Numeric ID of the List Header. Its value is always 0.

ROM / RAM Flag

This attribute, when TRUE, indicates that modifications to the Object Definitions are allowed.

Max Name Length

This attribute specifies the maximum length in octets for names of objects defined for this AP.

Access Protection Supported

This attribute, when TRUE, indicates that Access Protection is supported.

Version Of List

This attribute indicates the current version of this list. Each update to the list of object definitions, either locally initiated or through the use of the SetAttributes service causes the version number to be incremented.

List Local Reference

This attribute is a reference to the list that has local significance, such as an address. The value FFFFFFFF hex indicates that no local reference is available.

XXX Object Class Supported Constraint

This constraint selects object classes specified by XXX. The attributes following the constraint apply to Object Definitions of the selected object class.

XXX Numeric ID

This attribute is the Numeric ID of the first object in a series of objects of the class(es) selected by the constraint. All objects of the selected class(es) are contained in the series. The first Numeric ID for Data Types is always 1.

Number of XXX Entries

This attribute specifies the number of objects in the series.

XXX Local Reference

This attribute is a reference to the beginning of object definitions that describe the objects in the series. The reference has local significance, such as an address. The value FFFFFFFF hex indicates that no local reference is available.

List of Supported APO Classes and Objects

This attribute contains the Object Definitions maintained by the AP.

List of DLSAP Addresses

This attribute specifies the available DLSAP-Addresses usable by the AP to locate its AREPs.

NOTE DLSAP Address structure is defined in IEC 61158-3 and IEC 61158-4.

List of APO Classes Supported

This attribute specifies the classes of APOs supported by the AP.

List of Event Summary Selection Criteria

This optional attribute specifies a set of Boolean selection criteria that the AP is capable of using to select events when performing event summary processing. A set of selection criteria is specified by this standard. When an AP is defined, criteria specified for this attribute may be selected from this set, or they may be defined specifically for the AP. This list of criteria specifies

ENABLED	When TRUE, event reporting for event objects is enabled
	When FALSE, event reporting for event objects is disabled
DETECTED	When TRUE, event occurrence has been detected
	When FALSE, event occurrence has not been detected
ACKED	When TRUE, event acknowledgment has been received
	When FALSE, event acknowledgment is outstanding
ACTIVE	When TRUE, event detection is active for the event object
	When FALSE, event detection is not active for the event object

NOTE Additional values may be specified by this standard in the future or they may be user defined.

7.2.2.2.1.4 Supported Attributes

List of Supported Attributes

This optional attribute specifies the attributes supported by the object. This list contains, at a minimum, the mandatory attributes for the class of the object.

7.2.2.2.1.5 Services

Subscribe

This optional service is used by push subscribers to indicate to push publishers their requirements for receiving published data.

Identify

This optional service is used to request manufacturer information from the AP.

Get Status

This optional service is used to request the network visible state from the AP.

Status Notification

This optional service is used by the AP to report its network visible state.

Initiate

This optional service is used to open the AP context for an Application Relationship.

Terminate

This optional service is used to abruptly close the AP context for an Application Relationship.

Conclude

This service is used to gracefully close the AP context for an Application Relationship.

Reject

This service is initiated internally by the AP ASE to indicate that a confirmed service request APDU has been received with a protocol error.

7.2.2.3 Application Process ASE Service Specification

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

Subscribe

Identify

Get Status

Status Notification

Initiate

Terminate

Conclude

Reject

7.2.2.3.1 Subscribe Service

This confirmed service is used by a push subscriber wishing to dynamically request a push publisher for the publication of selected data. The service response contains sufficient information for the subscriber to assign an AREP to receive the data.

Upon receiving a Subscribe service indication, a push publisher may initiate the publication of an object, or modify the publication constraints of a published object. The actions and policies taken by an AP to satisfy a Subscribe indication are determined by the AP itself, and are outside the scope of this standard.

This service is also used by a push subscriber to indicate to the push publisher that it is no longer subscribing to the published object.

NOTE 1 There may or may not be other subscribers who may also receive the requested published data.

NOTE 2 Publishers do not necessarily need to receive such a request to publish. For example, a publisher could be pre-configured with the knowledge of what to publish.

Withdrawing
IECNORM.COM: Click to view the full PDF of IEC 61158-5:2000

7.2.2.3.1.1 Service Primitives

The service parameters for this service are shown in Table 9.

Table 9 – Subscribe Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
Application Relationship	M	M(=)		
Invoke ID	M	M(=)		
Joining	S	S(=)		
Published Object	M	M(=)		
Schedule Interval	M	M(=)		
Maximum ARs	U	U(=)		
Maximum Subscriber AP Permissible	U	U(=)		
Jitter				
Desired Publishing Start Time	U	U(=)		
Earliest Publishing Start Time	U	U(=)		
Latest Publishing Start Time	U	U(=)		
Leaving	S	S(=)		
List Of Leaving AREPs	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Data Type			C	C(=)
Data Length			C	C(=)
Dedicated			C	C(=)
List Of AR Information			C	C(=)
Publishing AREP ID			M	M(=)
DL Mapping Reference			M	M(=)
Scheduled Start Time			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Joining

This parameter is selected when an AP wants to subscribe to a published object. The Subscribe service is sent by the potential subscriber, using the selected AR, to the publisher of the published object.

Published Object

This parameter specifies one of the key attributes of the published object.

Schedule Interval

This parameter specifies the increment in the Fieldbus time between successive starting times for the scheduled sequence. The time granularity is the same as that defined for the underlying Data Link Layer (see IEC 61158-3).

Maximum ARs

This optional parameter specifies the maximum number of ARs on which the subscriber is willing to receive data. The default value for this parameter is one (1).

NOTE In certain circumstances, a Publisher AP may be requested to publish the same object on more than one AR. A subscriber AP may receive the same published object from a Publisher AP on more than one AR. The AR on which a Subscribe service is conducted may be different from the AR used to publish the data.

Maximum Subscriber AP Permissible Jitter

This optional parameter specifies the single upper bound on the two definitions of scheduling jitter on which the subscriber AP is willing to receive the data:

- a) the difference in starting times of a given scheduled sequence within successive subscriber AP cycles, and,
- b) the difference in starting times of a given subscriber AP cycle relative to a DLL cycle.

The value of this parameter is always less than the value of the Cycle Interval parameter. If this parameter is not specified, the jitter will be determined by the publisher. The granularity may be the same as defined for the Data Link Layer (see IEC 61158-3).

Desired Publishing Start Time

This optional parameter specifies the Fieldbus time at which publishing should start.

Earliest Publishing Start Time

This optional parameter specifies the earliest Fieldbus time at which publishing may start.

Latest Publishing Start Time

This optional parameter specifies the latest Fieldbus time at which the publishing is to start.

Leaving

This parameter is to be selected if a subscriber wants to stop its subscription to a specific object through a given AREP. The Subscribe service is sent to the Publisher AP using the AR used for joining.

List of Leaving AREPs

This parameter specifies the push publisher's AREP(s) that the subscribing AP wants to leave.

Result(+)

This selection type parameter indicates that the service request succeeded.

Data Type

This conditional parameter identifies the data type of the variable. This parameter is present if the subscriber is attempting to join a relationship.

Data Length

This conditional parameter specifies the length in octets of the data value to be published. This parameter is present if the subscriber is attempting to join a relationship and if the length is not defined for the data type of the published object.

Dedicated

This conditional parameter, when TRUE, indicates that the publishing relationship is dedicated. This parameter is present if the subscriber is attempting to join a relationship.

List of AR Information

This conditional parameter specifies the information necessary for the subscriber to construct its local subscriber AREP(s). This parameter is present if the subscriber is attempting to join a relationship. If present, this list is not empty.

Publishing AREP ID

This parameter specifies the numeric identifier for the AREP used for publishing:

DL Mapping Reference

This parameter identifies the DL mapping for the publishing AREP. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

Scheduled start time

This parameter specifies the actual publication start time.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.2.3.1.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

If the request indicates that a subscriber wishes to "join", the publishing AP locally determines if it can accommodate the specifics of the request. If it can, it may (1) reconfigure existing AR endpoint(s), or (2) allocate resources and create a new publishing AR endpoint(s), or (3) allocate bandwidth in the network schedule if necessary.

If the request indicates that a subscriber wishes to "leave", the AP locally determines if it can cease publishing over the indicated AR and still accommodate its other existing subscribers. If so, the AP may locally free up the AREP(s) and cancel the related network schedule(s).

If a confirmation (+) was returned for a "join" operation, the FAL user in the AP receiving the response APDU may configure or may create its local AREP(s) that will be used to receive the published variable.

If a confirmation (+) was returned for a "leave" operation, the FAL user in the AP may locally free up the resources associated with the AREP(s) used for subscription

NOTE It is a local matter for the FAL user to determine if the schedule specified by the publisher is sufficient. If insufficient, the local user should issue a Subscribe service primitive to inform the publisher that it is leaving.

7.2.2.3.2 Identify Service

Information to identify an AP is read with this service.

NOTE The attribute ProfileNumber of the AP is transmitted with the Initiate service.

7.2.2.3.2.1 Service Primitives

The service parameters for this service are shown in Table 10.

Table 10 – Identify

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Vendor Name			M	M(=)
Model Identifier			M	M(=)
Vendor Revision			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Vendor Name

This parameter contains the value of the Vendor Name attribute of the AP.

Model Identifier

This parameter contains the value of the Model Identifier attribute of the AP.

Vendor Revision

This parameter contains the value of the Vendor Revision attribute of the AP.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.2.3.2.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.2.3.3 Get Status Service

The device/user state is read with the Get Status service.

7.2.2.3.3.1 Service Primitives

The service parameters for this service are shown in Table 11.

Table 11 – Get status

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Service Access Status			M	M(=)
Operational Status			M	M(=)
Local Detail			U	U(=)
Result(-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service Access Status

This parameter contains the value of the Service Access Status attribute of the AP.

Operational Status

This parameter contains the value of the Operational Status attribute of the AP.

Local Detail

This parameter contains the user defined state of the application.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.2.3.3.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.2.3.4 Status Notification Service

The Status Notification service is used by the AP to report its state spontaneously.

7.2.2.3.4.1 Service Primitives

The service parameters for this service are shown in Table 12.

Table 12 – Status notification

Parameter name	.req	.ind
Argument	M	M(=)
AREP	M	M(=)
Destination DL-Address	C	
Source DL-Address		C
Service Access Status	M	M(=)
Operational Status	M	M(=)
LocalDetail	U	U(=)

Argument

The argument contains the parameters of the service request.

Destination DL-Address

This parameter exists only when the corresponding AREP supports it and the Remote Address Configuration Type is FREE. It gives the remote address to which the Status Notification should be sent.

Source DL-Address

This parameter exists only when the corresponding AREP supports it and the Remote Address Configuration Type is FREE. This parameter indicates the remote address from which the indicated Status Notification is to be sent.

Service Access Status

This parameter contains the value of the Service Access Status attribute of the AP.

Operational Status

This parameter contains the value of the Operational Status attribute of the AP.

Local Detail

This parameter contains the user defined state of the application.

7.2.2.3.4.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

7.2.2.3.5 Initiate Service

This service is used to establish a context between communicating APs for the exchange information on a single AR. The Initiate service negotiates the supported FAL services, the maximum outstanding services permitted, the maximum PDU length and the current version of the Object Definitions. The data structure nesting level is also negotiated if the local and remote APs support data structure nesting levels greater than one. The supported FAL services, the maximum outstanding services permitted, the maximum PDU length and data structure nesting level are AP system management attributes configured for the AREP. This service may be used to dynamically create the AREP to be used if the AREP does not exist.

7.2.2.3.5.1 Service Primitives

The service parameters for this service are shown in Table 13.

Table 13 – Initiate

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Version Object Definitions Calling	M	M(=)		
AP Descriptor Calling	M	M(=)		
Access Protection Supported Calling	M	M(=)		
Password Calling	M	M(=)		
Access Groups Calling	M	M(=)		
Destination DL-Address	C			
DL-Mapping	C	C		
Nesting Level Calling	U	U(=)		
User Detail	U	U(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Version OD Called			M	M(=)
AP Descriptor Called			M	M(=)
Access Protection Supported Called			M	M(=)
Password Called			M	M(=)
Access Groups Called			M	M(=)
DL-Mapping			C	C
Nesting Level Called			U	U(=)
User Detail			U	U(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Code			M	M(=)
Max PDU Length Sending Called				C
Max PDU Length Receiving Called				C
FAL Features Supported Called				C

Argument

This parameter carries the parameters of the service invocation.

Version Object Definitions Calling

This parameter specifies the version of the client's Object Definitions. Its value is null if the client does not contain Object Definitions.

AP Descriptor Calling

This parameter contains the value of the calling AP's Descriptor Reference attribute if it has one. If it does not, its value is null.

Access Protection Supported Calling

This parameter contains the value of the calling AP's Access Protection Supported attribute if it has one. If it does not, its value is null.

Password Calling

This parameter specifies the password to be used for the access to all objects of the server on this AR. If access with password is not to be used on this AR, its value is null.

Access Groups Calling

This parameter specifies the client's membership in specific access groups. The membership applies for the access to all objects of the server on this AR.

Destination DL-Address

This parameter exists only when the corresponding AR supports it and the Remote Address Configuration Type is FREE. It gives the remote address to which the connection shall be established.

DL-Mapping

This conditional parameter contains the DL-Mapping attributes for the AREP to be opened. It is used only when the AREP to be opened is not defined.

Nesting Level Calling

This optional parameter indicates the nesting level supported by the requester.

User Detail

This optional parameter contains user information.

Result(+)

This selection type parameter indicates that the service request succeeded.

Version Object Definition Called

This parameter shall specify the version of the server's Object Definitions.

AP Descriptor Called

This parameter contains the value of the called AP's Descriptor Reference attribute.

Access Protection Supported Called

This parameter contains the value of the called AP's Access Protection Supported attribute if it has one. If it does not, its value is null.

Password Called

This parameter specifies the password to be used for the access to all objects of the client on this AR. If access with password is not to be used on this AR, its value is null.

Access Groups Called

This parameter specifies the server's membership in specific access groups. The membership applies for the access to all objects of the client on this AR.

DL-Mapping

This conditional parameter contains the negotiated DL-Mapping attributes for the AREP to be opened. It is used only when the AREP to be opened is not defined.

Nesting Level Called

This optional parameter indicates the nesting level supported by the responder.

User Detail

This optional parameter contains user information.

Result(-)

This selection type parameter indicates that the service request failed.

Error Code

This parameter shall provide the reason for the failure.

Reason	Meaning
Max FAL PDU Size Insufficient	The maximum PDU length is not sufficient for the communication.
Service Not Supported	The requested service or option is not supported by the server.
User Initiate Denied	The FAL user refuses to establish the connection.
Version Object Definition incompatible	The called and calling versions of the Object Definitions are not compatible.
Password Error	There is already an AR established with the same password, or the password is not valid.
AP Descriptor incompatible	The descriptor is not supported by the server.
Other	Reason other than any of those identified above.

Max PDU Length Sending Called

This parameter shall specify the maximum length of the FAL PDU to be sent that may be handled on this AR.

It shall be transmitted by the called FAL and shall only be part of the Initiate.cnf primitive.

Max PDU Length Receiving Called

This parameter shall specify the maximum length of the FAL PDU to be received that may be handled on this AR.

It shall be transmitted, if supported, by the called FAL and shall only be part of the Initiate.cnf primitive.

FAL Services Supported Called

This parameter shall identify the optional AL services and the options supported by the server (see AR List).

It shall be transmitted, if supported, by the called FAL and shall only be part of the Initiate.cnf primitive.

7.2.2.3.5.2 Service Procedure

This service operates through a queue.

The requesting user submits the service request primitive to its FAL AE and starts an associated timer to monitor the request. The AP ASE builds the Initiate Request APDU Body and conveys it on the specified AR using the AR Establish Service request primitive. It also creates a transaction state machine.

Upon receipt of the Initiate Request APDU Body in an AR Establish service indication primitive, the responding AP ASE decodes it. If a protocol error was encountered while decoding the received APDU Body, the ASE uses the AR Abort Service to abort the AR. If a protocol error did not occur, the ASE delivers an Initiate service indication primitive to its user.

If the responding user is able to successfully process the request, the user returns an Initiate service response (+) primitive.

If the responding user is unable to successfully process the request, the service fails and the user issues an Initiate service response (-) primitive indicating the reason.

The responding AP ASE builds an Initiate service response APDU Body for a response (+) primitive or an Initiate service error APDU Body for a response (-) primitive and conveys it on the specified AR using an AR Establish service response primitive.

Upon receipt of the returned APDU Body in an AR Establish service confirmation (+ or -) primitive the initiating ASE delivers an Initiate service confirmation primitive to its user which specifies success or failure, and the reason for failure if a failure occurred. It also cancels the corresponding transaction state machine.

However, if the AP's transaction timer expires before the corresponding response or error APDU is received, the AP may take corrective actions, which may include instructing its local ASE to cancel the transaction state machine. Such instruction to the ASE would occur through a locally defined interface.

7.2.2.3.6 Terminate Service

This service shall be used to release an existing AR between APs, or to terminate a subscriber/receiver AREP's involvement in an AR.

7.2.2.3.6.1 Service Primitives

The service parameters for this service are shown in Table 14.

Table 14 – Terminate

Parameter name	.req	.ind
Argument	M	M
AREP	M	M(=)
Locally Generated		M
Terminate Identifier	M	M(=)
Reason Code	M	M(=)
Terminate Detail	U	U(=)

Argument

This parameter carries the parameters of the service invocation.

Locally Generated

This parameter shall indicate whether the termination was generated locally or by the communication partner.

The value false is not allowed if the Terminate Identifier parameter has the value FAL and the Reason Code parameter has the value AR Error.

Terminate Identifier

This parameter shall indicate where the reason for the termination has been detected. Possible values are:

FAL USER
FAL APO ASE
FAL AR ASE
DLL

Reason Code

This parameter shall specify the reason for the termination.

If the Terminate Identifier parameter has the value USER, the following values shall apply.

Reason	Meaning
Disconnection	The connection is released by the FAL user.
Version Object Definition incompatible	The called and calling versions of the Object Definitions are not compatible.
Password Error	There is already an AR established with the same password, or the password is not valid.
Descriptor incompatible	The server's descriptor is not supported by the client.
Limited Services Permitted	The AP is in the Logical Status LIMITED-SERVICES-PERMITTED.

If the Abort Identifier parameter has the value FAL, the following values shall apply:

Reason	Meaning
AR Error	Faulty AR
User Error	Improper, unknown or faulty service primitive received from the FAL user
FAL PDU Error	Unknown or faulty FAL PDU received from the AR ASE
Connection State Conflict AR ASE	Improper AR ASE service primitive
AR ASE Error	Unknown or faulty AR ASE service primitive
PDU Size	PDU length exceeds maximum PDU length
Feature Not Supported	SERVICE-REQ PDU received from the AR ASE and service or option not supported as a server (see FAL Features Supported attribute in the AR)
Invoke ID Error Response	Confirmed service.rsp received from the FAL user and Invoke ID does not exist or CONFIRMED_SERVICE-RSP_PDU received from the AR ASE and Invoke ID does not exist
Max Services Overflow	CONFIRMED_SERVICE-REQ_PDU received from the AR ASE and Outstanding Services Counter Receiving $\geq$ Max Outstanding Services Receiving
Connection State Conflict Service Error	INITIATE-REQ_PDU received from the AR ASE The service in the response does not match the service in the indication or the service in the confirmation does not match the service in the request.
Invoke ID Error Request	CONFIRMED_SERVICE-REQ_PDU received from the AR ASE and Invoke ID already exists.

If the Terminate Identifier parameter has the value AR ASE or DLL, the Reason Code parameter shall be supplied by the AR ASE.

Terminate Detail

This optional parameter contains additional information about the abort reason (max. 16 octets). In case of error reports from the application, the meaning is defined in the profile.

7.2.2.3.6.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

7.2.2.3.7 Conclude Service

This service shall be used to gracefully release an existing AR between APs.

7.2.2.3.7.1 Service Primitives

The service parameters for this service are shown in Table 15.

Table 15 – Conclude

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 Info			M	M(=)

Argument

This parameter carries the parameters of the service invocation.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.2.3.7.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.2.3.8 Reject Service

This service is used to inform the remote endpoint that a protocol error has been detected. It is generated by the AP ASE if an APDU has been received with an invalid service type code. When another APO ASE detects a protocol error, it internally requests the AP ASE to generate a Reject PDU. This service is not used to indicate that a service error has been detected by the user.

7.2.2.3.8.1 Service Primitives

The service parameters for this service are shown in Table 16.

Table 16 – Reject

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Reject Code	M	M(=)
Original FAL Service Type	M	M(=)
Original Invoke ID	M	M(=)
Original PDU Body	U	U(=)

Argument

The argument contains the parameters of the service request.

Reject Code

This parameter indicates the reason for rejection. The values of this may indicate:

- Unsupported Service
- AR Not Established
- AR Not Defined
- Max Outstanding Requests Exceeded
- Max PDU Size Exceeded
- Duplicate Invoke ID

Original FAL Service Type

This parameter contains the service type of the rejected service request.

Original Invoke ID

This parameter contains the invoke id of the rejected service request.

Original PDU Body

This optional parameter contains some or all of the data of the APDU that was rejected. The amount of data included is determined by the user.

7.2.2.3.8.2 Service Procedure

The Reject service is a service that operates through a queue or buffer. It may be initiated only by the FAL AP ASE.

If any APO ASE receives a confirmed request APDU that contains a protocol error, the AP ASE builds a Reject Request APDU and returns it on the specified AR.

Upon receipt of the Reject Request APDU, the receiving AP ASE delivers an AR-Reject.indication primitive through the FAL ASE that issued the confirmed send request primitive, and it issues a negative confirmation to the requesting user.

7.2.3 Application Relationship ASE

7.2.3.1 Overview

In a distributed system, application processes communicate with each other by exchanging application layer messages across well-defined application layer communications channels. These communication channels are modeled in the Fieldbus Application Layer as application relationships (ARs).

ARs are responsible for conveying messages between applications according to specific communications characteristics required by time-critical systems. Different combinations of these characteristics lead to the definition of different types of ARs. The characteristics of ARs are defined formally as attributes of AR Endpoint classes.

The messages that are conveyed by ARs are FAL service requests and responses. Each is submitted to the AR ASE for transfer by an FAL Application Service Element (ASE) that represents the class of the APO being accessed. Figure 16 illustrates this concept.

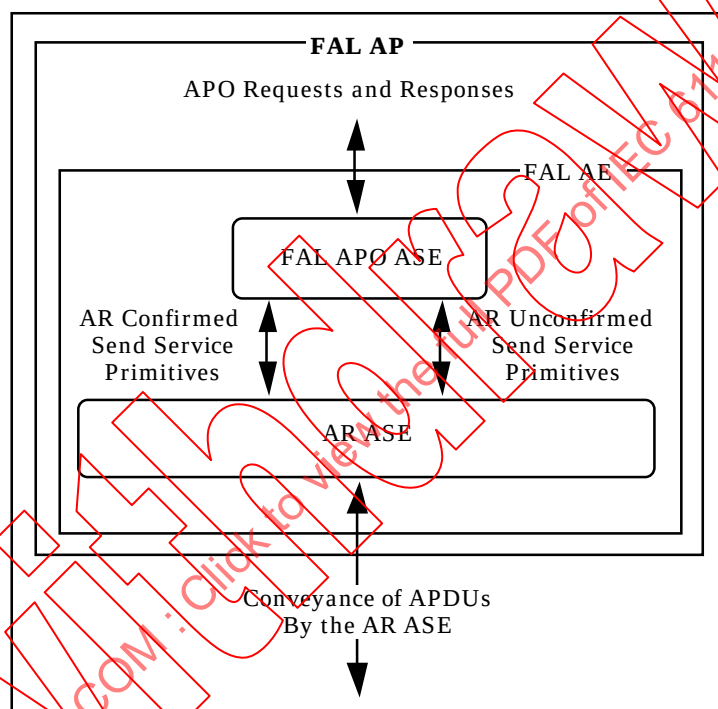


Figure 16 – The AR ASE Conveys APDUs between APs

Depending on the type of AR, APDUs may be sent to one or more destination application processes connected by the AR. Other characteristics of the AR determine how APDUs are to be transferred. These characteristics are described below.

7.2.3.1.1 Endpoint Context

Each AP involved in an AR contains an endpoint of the AR. Each AR endpoint is defined within the AE of the AP. Its definition, when combined with the definitions of the other endpoints defines an AR. To ensure communications compatibility among or between endpoints, each endpoint definition contains a set of compatibility-related characteristics. These characteristics must be configured appropriately for each endpoint for the AR to operate properly.

Endpoint definitions also contain a set of characteristics that describe the operation of the AR. These characteristics, when combined with those used to specify compatibility, define the context of the endpoint. The endpoint context is used by the AR ASE to manage the operation of the endpoint and the conveyance of APDUs. The characteristics which comprise the endpoint context are described next.

7.2.3.1.1.1 Endpoint Role

The role of an AREP determines the permissible behavior of an AP at the AREP. An AREP role may be that of a client, server, peer (client and/or server), push or pull publisher, push or pull subscriber, or push or pull publishing manager.

NOTE See concepts clause for description of clients and servers, and the push and pull models for publishers and subscribers.

Table 17 and Table 18 summarize the characteristics and combinations of each of the AREP roles.

Table 17 – Conveyance of Service Primitives by AREP Role

	Client	Server	Peer	Push Publisher	Push Subscriber	Pull Publ Mgr	Pull Publisher	Pull Subscriber
Send Service Req	X		X	X		X		
Recv Service Req		X	X		X		X	
Send Service Rsp		X	X				X	
Recv Service Rsp	X		X			X		X

Table 18 – Valid Combinations of AREP Roles Involved in an AR

AREP Roles by Interaction Model	Client	Server	Peer	Push Publisher	Push Subscriber	Pull Publ Mgr	Pull Publisher	Pull Subscriber
Client / Server		Yes	Yes					
Client / Peer			Yes					
Server / Peer			Yes					
Publisher / Subscriber								
Push Publisher					Yes			
Push Subscriber								
Pull Publ Mgr							Yes	
Pull Publisher								Yes
Pull Subscriber								

7.2.3.1.1.2 Dedicated AR Endpoints

AR endpoints that are dedicated provide their services directly to the FAL User. Although their behavior is the same as non-dedicated endpoints, the APDUs they convey contain no service specific protocol control information other than the AR control field.

FAL Users configured for dedicated ARs construct and transfer their data without using the services of the other FAL ASEs. The format and contents of the user data conveyed over dedicated ARs are known only through configuration.

7.2.3.1.1.3 Cardinality

The cardinality of an AR specifies, from the point of view of a client or publisher endpoint, how many remote application processes are involved in an AR. Cardinality is never expressed from the viewpoint of a server or a subscriber.

When expressed from the viewpoint of a client or peer endpoint, ARs are always 1-to-1. Clients are never capable of issuing a request and waiting for responses from multiple servers.

When expressed from the viewpoint of a publisher endpoint, they indicate that multiple subscribers are supported. These ARs are 1-to-many. ARs that are 1-to-many provide for communications between one application and a group of (one or more) applications. They are

often referred to as multi-party and multi-cast. To support the “pull” model of publishing, these ARs provide a management path between the publishing manager and the publisher.

In 1-to-many ARs, the publisher endpoint identifies the remote endpoints using a group identifier, rather than identifying each one individually as is done with 1-to-1 ARs. This permits subscribers to join and leave ARs without disrupting the publisher endpoint context because their individual identities are not known to the publisher endpoint. However, the publisher application may be aware of their participation, but that information is not part of the AR endpoint context.

7.2.3.1.1.4 Conveyance Model

The conveyance model defines how APDUs are sent between endpoints of an AR. Three characteristics are used to define these transfers:

- conveyance paths
- trigger policy, and
- conveyance policy.

7.2.3.1.1.4.1 Conveyance Paths

The purpose of AR ASEs is to transfer information between AR endpoints. This information transfer occurs over the conveyance paths of an AR. A conveyance path represents a one-way communication path used by an endpoint for input or output.

To support the role of the application process, the endpoint must be configured with either one or two conveyance paths. Endpoints that only send, or only receive, are configured with either a send or receive conveyance path, and those that do both are configured with both. ARs with a single conveyance path are called uni-directional, and those with two conveyance paths are called bi-directional.

Uni-directional ARs are capable of conveying service requests only. To convey service responses, a bi-directional AR is necessary. Therefore, uni-directional ARs support the transfer of unconfirmed services in one direction only, while bi-directional ARs support the transfer of unconfirmed and confirmed services initiated by only one endpoint, or by both endpoints.

7.2.3.1.1.4.2 Trigger Policy

Trigger policy indicates when APDUs are transmitted by the Data Link Layer over the network.

The first type is referred to as user-triggered. User-triggered AREPs submit FAL APDUs to the Data Link Layer for transmission at its earliest opportunity.

The second type is referred to as network-scheduled. Network scheduled AREPs submit FAL APDUs to the Data Link Layer for transmission according to a schedule configured by management.

This network scheduling mechanism can be either cyclic or one-time.

Network-scheduled ARs may also convey requests and responses asynchronously if the underlying layer has available bandwidth through use of the compel service. These transfers occur in addition to the scheduled transfers, and without being triggered from the network schedule.

7.2.3.1.1.4.3 Conveyance Policy

Conveyance policy indicates whether APDUs are transferred according to a buffer model or a queue model. These models describe the method of conveying APDUs from sender to receiver.

Buffered ARs contain conveyance paths that have a single buffer at each endpoint. Updates to the source buffer are conveyed to the destination buffer according to the trigger policy of the AR. Updates to either buffer replace its contents with new data. In buffered ARs, unconveyed or undelivered data that has been replaced is lost. Additionally, data contained in a buffer may be read multiple times without destroying its contents.

Queued ARs contain conveyance paths that are represented as a queue between endpoints. Queued ARs convey data using a first-in first-out (FIFO) queue. Queued ARs are not overwritten; new entries are queued until they can be conveyed and delivered.

If a queue is full, a new message will not be placed into the queue.

NOTE The AR conveyance services are described abstractly in such a way that they are capable of being implemented to operate using buffers or queues. These services may be implemented in a number of ways. For example, they may be implemented such that the capability is provided to load the buffer/queue, and subsequently post it for transfer by the underlying Data Link Layer. Or, these services may be implemented such that these capabilities are combined so that the buffer/queue may be loaded and transferred in a single request. On the receiving side, these services may be implemented by delivering the data when it is received, or by indicating its receipt and allowing the user to retrieve it in a separate operation. Another option is to require the user to detect that the buffer or queue has been updated.

7.2.3.1.2 Underlying Communications Services

The AR ASE conveys FAL APDUs using the capabilities of the underlying Data Link Layer. Several characteristics are used to describe these capabilities. This subclause provides a description of each. These characteristics are specific to the Data Link Mapping defined in IEC 61158-6. Their precise specification can be found there.

7.2.3.1.2.1 Connectionless and Connection-oriented Services

The underlying layer may support AR endpoints by providing connectionless or connection-oriented services. Endpoints configured to operate over connection-oriented services may be capable of opening and closing connections if the connections they use are not pre-configured to be open.

NOTE Connectionless and connection-oriented are DL terms. Refer to IEC 61158-3 and IEC 61158-4 for definition.

7.2.3.1.2.2 Buffered and Queued Services

The underlying layer may support AR endpoints by providing buffered or queued services. These services can be used to implement buffers or queues required by certain classes of endpoints.

7.2.3.1.2.3 Cyclic and Acyclic Transfers

The underlying layer may support AR endpoints by providing cyclic or acyclic services.

The underlying layer may also provide for the acyclic transfer of APDUs concurrently with cyclic transfer of APDUs between the same endpoints.

7.2.3.1.2.4 Quality of Service

The underlying layer may provide more than one quality of service. When this occurs, the AREP must be configured to use the quality of service appropriate for the APDUs it conveys. Quality of service parameters for AREPs are contained in the data link mappings in IEC 61158-6.

7.2.3.1.2.5 Segmentation

The FAL has the capability of segmenting user data conveyed by it. In addition, when the underlying layer provides segmentation, too, the maximum length of data conveyed by the FAL in a single APDU will increase accordingly.

7.2.3.1.2.6 Address Format

The underlying layer may support more than one address format. When it does, all endpoints of an AR must be mapped to compatible address formats. DL-mappings for AREPs are specified in IEC 61158-6 for the Data Link Layer (IEC 61158-3).

7.2.3.1.2.7 Timeliness

AR endpoints may be defined to support timeliness information about their transfers as calculated by the Data Link Layer. This timeliness information is received from the Data Link Layer and forwarded to the FAL user as a parameter of the associated FAL service. Publisher and subscriber AR endpoints support transparent, residence, synchronized, and update timeliness. These types of timeliness are described in the definitions of each of these endpoint types below. The mechanism used by the Data Link Layer to calculate timeliness of published data is defined in IEC 61158-3 and IEC 61158-4. The attributes defined by the AR Model to support timeliness are defined in the DLL Mapping Attributes for the Publisher/Subscriber AREPs in IEC 61158-6.

7.2.3.1.2.8 Duplicate FAL PDU detection

AR endpoints may be defined to support detection of the receipt of duplicate FAL PDUs. Alternatively, an indication may be received from the Data Link Layer to support this capability and forwarded to the FAL user as a parameter of the associated FAL service. The attributes defined by the AR Model to support this capability are defined in the DLL Mapping Attributes in IEC 61158-6.

7.2.3.1.3 AR Establishment

For an AR endpoint to be used by an application process, the corresponding AR has to be active. When an AR is activated, it is referred to as "established".

AR establishment can occur in one of three ways. First, ARs can be pre-established. Pre-established means that the AE that maintains the endpoint context is created when the AP is connected to the network. In this case, communications among the applications involved in the AR may take place without first having to explicitly establish the AR. Any AR may be defined to be pre-established.

Second, ARs can be pre-defined, but not pre-established. Pre-defined means that the characteristics of the AR are known to each endpoint, but that their contexts have not been synchronized for operation. In this case, the use of the FAL Establish service is required to synchronize them and open them for data transfer.

Third, ARs can require dynamic definition and establishment. In this case, definitions have to be created for each of the AREPs using the Create service. After creation, they are established using the Establish service if they were not created as "established".

7.2.3.1.4 Application Relationship Classes

AREPs are defined with a combination of characteristics to form different classes of ARs.

7.2.3.2 Application Relationship Endpoint Class Specifications

7.2.3.2.1 AR Endpoint Formal Model

The AR endpoint formal model defines the characteristics common to all AR endpoints. this class is not capable of being instantiated. It is present only for the inheritance of its attributes and services by its subclasses, each specified in a separate subclause of this standard.

All AR endpoint attributes are accessible through system Management (see the future IEC 61158-7), and not through the Object Management ASE services defined in this standard.

FAL ASE: **AR ASE**
CLASS: AR ENDPOINT
CLASS ID: 32
PARENT CLASS: TOP

SYSTEM MANAGEMENT ATTRIBUTES:

1	(m)	Attribute:	Local AP
2	(m)	Attribute:	FAL Revision
3	(m)	Attribute:	Dedicated (TRUE, FALSE)
4	(o)	Attribute:	Transfer Syntax
5	(o)	Attribute:	List Of Supported Attributes

SERVICES:

1	(o)	OpsService:	AR-Unconfirmed Send
2	(o)	OpsService:	AR-Confirmed Send
3	(o)	OpsService:	AR-Establish
4	(o)	OpsService:	AR-DeEstablish
5	(o)	OpsService:	AR-Abort
6	(o)	OpsService:	AR-Compel
7	(o)	OpsService:	AR-Get Buffered Message
8	(o)	OpsService:	AR-Schedule Communication
9	(o)	OpsService:	AR-Cancel Scheduled Sequence
10	(o)	OpsService:	AR-Get DL-Time
11	(o)	OpsService:	AR-Status
12	(o)	OpsService:	AR-XON-OFF
13	(o)	OpsService:	AR-RemoteRead
14	(o)	OpsService:	AR-RemoteWrite

7.2.3.2.1.1 System Management Attributes**Local AP**

This attribute identifies the AP attached or configured to use the AREP using a local reference.

FAL Revision

This specifies the revision level of the FAL protocol used by this endpoint. The revision level is in the AR header of all FALPDUs transmitted.

Dedicated

The attribute specifies, when TRUE, that the endpoint is dedicated. When TRUE, the services of the AR ASE are accessed directly by the FAL User.

Transfer Syntax

This optional attribute identifies the encoding rules to be used on the AR. When not present, the default FAL Transfer Syntax of this standard is used.

List of Supported Attributes

This optional attribute specifies the attributes supported by the object. This list contains, at a minimum, the mandatory attributes for the class of the object.

7.2.3.2.1.2 Services

All services defined for this class are optional. When an instance of the class is defined, at least one has to be selected.

AR-Unconfirmed Send

This optional service is used to send an unconfirmed service.

AR-Confirmed Send

This optional service is used to send a confirmed service.

AR-Establish

This optional service is used to establish (open) an AR.

AR-DeEstablish

This optional service is used by client or peer users to request the graceful termination of a 1-to-1 AR.

AR-Abort

This optional service is used to abort (abruptly terminate) an AR.

AR-Compel

This optional service is used by the FAL user to request the AR ASE to convey a message which has been deferred until explicitly released.

AR-Get Buffered Message

This optional service is used to request the AR to retrieve a message which is being maintained in a buffer in the local Data Link Layer.

AR-Schedule Communication

This optional service is used to schedule a sequence of DL-COMPELs for a particular relationship.

AR-Cancel Scheduled Sequence

This optional service is used to cancel an existing sequence which has previously been scheduled.

AR-Get DL-Time

This optional service is used to access the DL-Time service of the Data Link Layer.

AR-Status

This optional service is used to indicate to the AR user that an internal event of the AREP, such as the transmission of a network-triggered buffer, has occurred.

AR-XON-OFF

This optional service causes the flow of data on a specified AR to be suspended or resumed.

AR-RemoteRead

This optional service is used to request the AR to retrieve a message which is being maintained in a buffer in the remote Data Link Layer.

AR-RemoteWrite

This optional service is used to trigger a data exchange, after writing a value in a local buffer.

7.2.3.3 Application Relationship ASE Service Specifications

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

- AR-Unconfirmed Send
- AR-Confirmed Send
- AR-Establish
- AR-DeEstablish
- AR-Abort
- AR-Compel
- AR-Get Buffered Message
- AR-Schedule Communication
- AR-Cancel Scheduled Sequence
- AR-Status
- AR-XON-OFF
- AR-RemoteRead
- AR-RemoteWrite

The services AR-Confirmed Send, AR-Establish, and AR-DeEstablish contain the FAL PDU Body as part of the Result parameter in the response and confirmation primitives. The FAL PDU Body may contain either the positive or negative responses returned by the FAL User transparently to the AR ASE. Therefore, these services have a single Result parameter instead of separate Result(+) and Result(-) parameters that are commonly used to convey the positive and negative responses returned by the FAL User.

7.2.3.3.1 AR-Unconfirmed Send Service

This service is used to send AR-Unconfirmed request APDUs for FAL APO ASEs. The AR-Unconfirmed Send service may be requested at either endpoint of a one-to-one bi-directional AR, at the server endpoint of a one-to-one uni-directional AR, or at the publisher endpoint of a 1-to-many AR.

NOTE This service is described abstractly in such a way that it is capable of operating with ARs that convey FAL APDUs through buffers or queues. This service may be implemented in such a way that the capability is provided to load the buffer/queue, and subsequently post it for transfer by the underlying Data Link Layer. Alternatively, this service may be implemented such that these capabilities are combined so that the user may load the buffer/queue and request its transfer in a single operation.

7.2.3.3.1.1 Service Primitives

The service parameters for this service are shown in Table 19.

Table 19 – AR-Unconfirmed Send

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Remote DLSAP Address	C	C
FAL Service Type	M	M(=)
FAL APDU Body	M	M(=)
Timeliness		C
Duplicate FAL PDU Body		C

Argument

The argument contains the parameters of the service request.

Remote DLSAP Address

This conditional parameter contains the destination DLSAP address in the request and the source DLSAP address in the indication. It is present if the AREP is configured with a Remote Address Configuration Type of FREE.

FAL Service Type

This parameter contains the type of the service being conveyed.

FAL APDU Body

This parameter contains the service dependent body for the APDU.

Timeliness

This conditional parameter indicates both the local and remote timeliness status of the FAL APDU Body contained in the AR-Unconfirmed Send request primitive. This parameter is present if Timeliness is supported in the DL Mapping Attributes of the AREP. The values associated with this parameter are defined in the description of the DLL Mapping Attributes in IEC 61158-6.

Duplicate FAL PDU Body

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the Data Link Layer. It is present if supported in the DL Mapping Attributes of the AREP.

7.2.3.3.1.2 Service Procedure

The AR-Unconfirmed Send Service is a service that operates through a queue or buffer.

The requesting FAL ASE submits an AR-Unconfirmed send.request primitive to its AR ASE. The AR ASE builds an AR-Unconfirmed Send request APDU. If the local AREP for the specified AR supports timeliness, the AR ASE indicates the publishing and transmission timeliness in the APDU.

If the AREP is queued, the AR ASE queues the APDU for submission to the lower layer. If the AR is buffered, the AR ASE replaces the previous contents of the buffer with the APDU contained in the service primitive.

If the AREP is user-triggered, the AR ASE immediately requests the lower layer to transfer the APDU. If the AR is network-scheduled, the AR ASE requests the DL to transfer the data at the scheduled time. The Data Link mapping indicates how the AR ASE coordinates its requests to transmit the data with the Data Link Layer.

NOTE The transmission schedule is managed by the underlying layer, not the AR-ASE. Refer to IEC 61158-3 and IEC 61158-4 for further details.

Upon receipt of the AR-Unconfirmed Send request APDU, the receiving AR ASE delivers an AR-Unconfirmed send.indication primitive to the appropriate FAL ASE as indicated by the FAL Service Type Parameter. If the receiving AREP supports timeliness, the AR ASE includes the timeliness parameters received from the Data Link Layer in the indication primitive.

7.2.3.3.2 AR-Confirmed Send Service

This service is used to send confirmed request and response APDUs for FAL APO ASEs. The AR-Confirmed Send service may be requested at Client and Peer endpoints of one-to-one bi-directional ARs.

NOTE This service is described abstractly in such a way that it is capable of operating with ARs that convey FAL APDUs through buffers or queues. This service may be implemented in such a way that the capability is provided to load the buffer/queue, and subsequently post it for transfer by the underlying Data Link Layer. Alternatively, this service may be implemented such that these capabilities are combined so that the user may load the buffer/queue and request its transfer in a single operation.

7.2.3.3.2.1 Service Primitives

The service parameters for this service are shown in Table 20.

Table 20 – AR-Confirmed send

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Destination DL-Address	C	C		
FAL Service Type	M	M(=)		
FAL APDU Body	M	M(=)		
Result			M	M(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Source DL-Address			C	C
FAL Service Type			M	M(=)
FAL APDU Body			M	M(=)
Timeliness				C

Argument

The argument contains the parameters of the service request.

Destination DL-Address

This conditional parameter contains the Destination DL-address. It is present only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE.

NOTE Not all confirmed services support this parameter. The use of this parameter when using confirmed services is a local matter.

FAL Service Type

This parameter contains the type of the service being conveyed.

FAL APDU Body

This parameter contains the service dependent body for the APDU.

Result

This selection type parameter indicates that the service request succeeded.

Source DL-Address

This conditional parameter contains the Source DL-address. It is present only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE.

NOTE Not all confirmed services support this parameter. The use of this parameter with confirmed services is a local matter.

FAL Service Type

This parameter contains the type of the service being conveyed.

FAL APDU Body

This parameter contains the service dependent body for the APDU.

Timeliness

This conditional parameter indicates the timeliness of the update into the FAL. This parameter is present if Timeliness is supported in the DL Mapping Attributes of the AREP. The values associated with this parameter are defined in the description of the DLL Mapping Attributes in IEC 61158-6.

7.2.3.3.2.2 Service Procedure

The AR-Confirmed Send Service is a service that operates through a queue or buffer.

The requesting FAL ASE submits an AR-Confirmed Send request primitive to its AR ASE. The AR ASE creates a transaction state machine to control the invocation of the service.

The AR ASE builds an AR-Confirmed Send request APDU. If the local AREP for the specified AR supports timeliness, the AR ASE indicates the publishing and transmission timeliness in the APDU.

If the AREP is queued, the AR ASE queues the APDU for submission to the lower layer. If the AR is buffered, the AR ASE replaces the previous contents of the buffer with the APDU contained in the service primitive.

If the AREP is user-triggered, the AR ASE immediately requests the lower layer to transfer the APDU. If the AR is network-scheduled, the AR ASE requests the DL to transfer the data at the scheduled time. The Data Link mapping indicates how the AR ASE coordinates its requests to transmit the data with the Data Link Layer.

NOTE The transmission schedule is managed by the underlying layer, not the AR-ASE. Refer to IEC 61158-3 and IEC 61158-4 for further details.

Upon receipt of the AR-Confirmed Send request APDU, the receiving AR ASE delivers an AR-Confirmed Send indication primitive to the appropriate FAL ASE as indicated by the FAL Service Type Parameter. If the receiving AREP supports timeliness, the AR ASE includes the timeliness parameters received from the Data Link Layer in the indication primitive.

The responding FAL ASE submits a confirmed send response primitive to its AR ASE. The AR ASE builds a confirmed send response APDU.

If the AREP is queued, the AR ASE queues the APDU for submission to the lower layer. If the AR is buffered, the AR ASE replaces the previous contents of the buffer with the APDU contained in the service primitive.

If the AREP is user-triggered, the AR ASE immediately requests the lower layer to transfer the APDU. If the AR is network-scheduled, the AR ASE requests the DL to transfer the data at the scheduled time. The data link mapping indicates how the AR ASE coordinates its requests to transmit the data with the Data Link Layer.

Upon receipt of the confirmed send response APDU, the receiving AR ASE uses the Invoke ID contained in the response APDU to associate the response with the appropriate request and cancel the associated transaction state machine. The AR ASE delivers an AR-Confirmed Send confirmation primitive to the requesting FAL ASE. If the receiving AREP supports timeliness, the AR ASE includes the timeliness parameters received from the Data Link Layer in the confirmation primitive.

If the timer expires before the sending AR ASE receives the response APDU, the AR ASE cancels the associated transaction state machine and delivers an AR-Confirmed Send confirmation (-) primitive to the requesting FAL ASE.

7.2.3.3.3 AR-Establish Service

This confirmed service operates in a pair-wise manner between two AR endpoints to synchronize their contexts and activate them for the transfer of APDUs.

The endpoint context may be created during establishment or prior to establishment through System Management. Once defined, the Set Attributes and Get Attributes services can be used to update and further coordinate endpoint contexts.

7.2.3.3.3.1 Service Primitives

The service parameters for this service are shown in Table 21.

Table 21 – AR-Establish Service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Remote DL Address	C	C(=)		
User Data	U	U(=)		
Result			M	M(=)
AREP			M	M(=)
User Data			U	U(=)

Argument

The argument contains the parameters of the service request.

AREP

This parameter contains sufficient information to locally identify the AREP to be established.

Remote DL Address

This conditional parameter identifies the remote DL address. If the AR to be established is supported by a DL-Connection, then the DL-Address is a DLCEP address. If not, the DL-Address is a DLSAP address. It is present when the AREP Remote Address Configuration Type attribute is FREE.

User Data

This optional parameter contains user supplied data that is to be conveyed with the service request.

Result

This parameter indicates that the service request succeeded or failed.

AREP

This parameter contains sufficient information to locally identify the AREP to be established.

User Data

This optional parameter contains user supplied data that is to be conveyed with the service response.

7.2.3.3.3.2 Service Procedure

7.2.3.3.3.2.1 1-to-1 AR Establishment

When used in support of 1-to-1 AREPs, the AR-Establish service causes Establish PDUs to be exchanged between calling and called DLSAPs as shown in Figure 17.

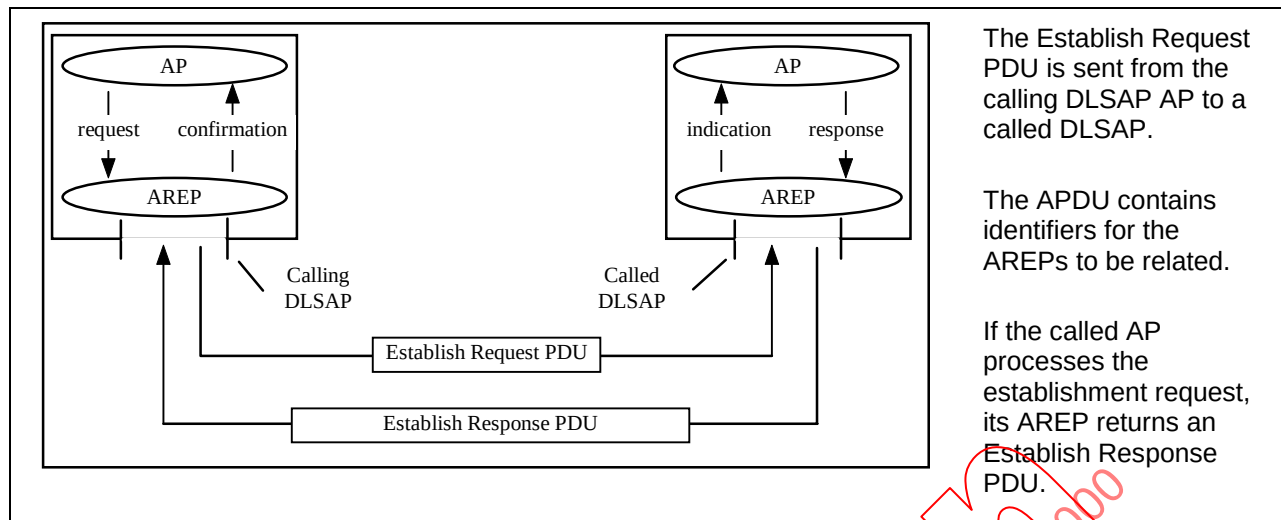


Figure 17 – 1-to-1 AR Establishment

Upon receipt of an AR-Establish request service primitive, the calling FAL issues an Establish Request APDU to the called AP ASE that handles establishment of an AR. Upon receipt of an Establish indication primitive, the called AP ASE examines the parameters specified in it and returns the appropriate response in the AR-Establish response primitive.

7.2.3.3.2.2 1-to-Many AR Establishment

When used in support of 1-to-Many AREPs, the AR-Establish service causes local AREPs to be established independently, as shown in Figure 18.

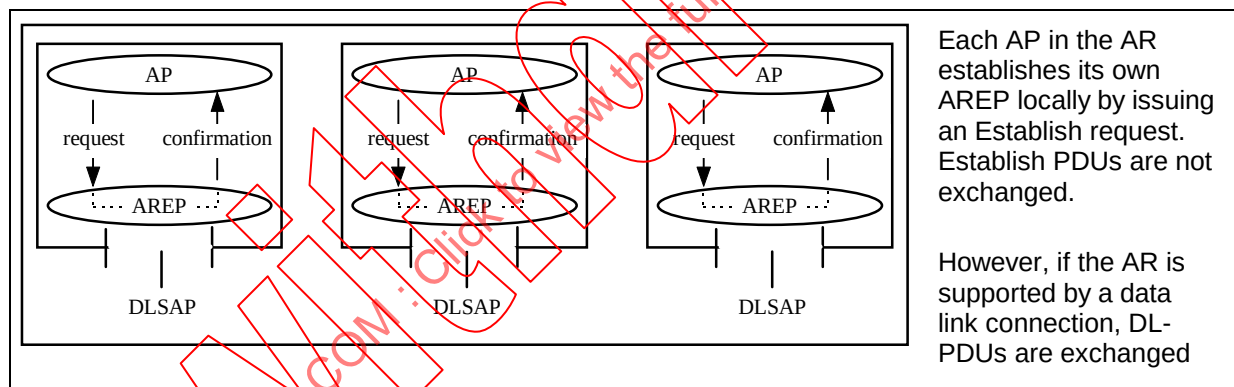


Figure 18 – 1-to-many AR establishment

Upon receipt of an AR-Establish request service primitive, the calling FAL issues an AR-Establish confirmation service primitive to the calling AP indicating whether or not it was able to establish the AREP. If one of the following cases is found, it was not able to establish the AREP:

- 1) the requested AREP is not specified within the FAL, or
- 2) resources are not available to establish the requested AREP, or
- 3) for AREPs supported by a data link connection, the Data Link Layer was not able to establish the data link connection.

7.2.3.3.2.3 Compatible AREP Classes

Table 22 provides possible combinations of the AREP classes which may be related with the AR-Establish service. In this table the SERVER column contains a "-" to indicate that Server AREPs do not initiate AR establishment.

Table 22 – Valid Combinations of AREP Classes to be Related

		Calling AREP Role		
		PEER	CLIENT	SERVER
Called AREP Role	PEER	YES	YES	--
	CLIENT	NO	NO	--
	SERVER	YES	YES	-

If the called AP decides to establish an AR, it issues an AR-Establish response primitive. The FAL returns the AR parameter which is used to refer to the AR being formed from the called service user. If the AR being established uses the Connection Oriented Data Link Layer and its explicit establishment is required, it is established before an Establish Response APDU is returned. The called AP then returns an Establish Response APDU with the Result(+) parameter to the calling AR_ASE. The calling AR_ASE issues an AR-Establish confirmation primitive in which the AR parameter is specified.

7.2.3.3.2.4 Conflict Resolution

The normal establishment of ARs follows the general time-sequence for confirmed services. In the cases where each endpoint of an AR concurrently issues an AR-Establish request, a conflict arises. The algorithms for resolving conflicts of this type are specified in detail in IEC 61158-6.

7.2.3.3.4 AR-DeEstablish Service

This confirmed service is invoked by client or peer users to request the graceful termination of a 1-to-1 application relationship. Use of the AR-DeEstablish service causes the endpoints of the AR to be closed. They may be reopened using the AR-Establish service. This service may be used on any open AREP, whether or not the AR was pre-established or dynamically established using the AR-Establish service.

When a request to de-establish an AR is made, the local AR ASE conveys the DeEstablish Request PDU to the remote endpoint of the AR, and accepts no additional request PDUs from the user unless an AR-Abort is requested. Upon receipt of a DeEstablish Response PDU, it clears all outstanding service requests, closes the endpoint context, and informs the user using the confirm service primitive.

The remote AR ASE informs its user when an AR-DeEstablish Request PDU is received using the indication primitive for the service. The user returns a response primitive after responding to the service indications it received. When the remote AR ASE receives the response, it closes the endpoint context, and returns an AR- DeEstablish Response PDU.

7.2.3.3.4.1 Service Primitives

The service parameters for this service are shown in Table 23. The use of this service is for future study.

Table 23 – AR-DeEstablish Service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.3.3.4.2 Service Procedure

The AR-DeEstablish Service is a service that operates through a queue or buffer. Its service procedure is for future study.

7.2.3.3.5 AR-Abort Service

The service is used by the FAL User to abruptly terminate an AR. It is always successful; the receiver of an Abort request or Abort request APDU always aborts the AR. This service may be used on any open AREP, whether or not the AR was pre-established or dynamically established using the establish service.

The AR-Abort Service is used to instruct the AR ASE to abruptly terminate all activity on an AREP and place it in the Closed state. Receipt of an AR-Abort request primitive causes the AR ASE to immediately close the AREP context and issue an Abort Request APDU to the remote AREPs. Receipt of an Abort Request APDU causes the AR ASE to immediately close the AR AREP context and deliver an AR-Abort request indication primitive to the user. The immediate close of each endpoint context causes all outstanding service requests to be cleared. All subsequent service primitives and APDUs received by the AR ASE for the aborted AR are discarded except for those of the AR-Establish service.

The AR-Abort service may be requested at either AREP of a one-to-one relationship, or at the publisher endpoint of a 1-to-many or 1-to-all AR. Subscribers are not capable of aborting ARs with publishers, although they may close their own endpoints through local means.

The AR ASE may also initiate the abort service when it detects unrecoverable communication failures. In this case, the AR ASE delivers an AR-Abort indication primitive informing the user of the failure and closes the endpoint context.

7.2.3.3.5.1 Service Primitives

The service parameters for this service are shown in Table 24.

Table 24 – AR-Abort

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Locally Generated		M
Originator	M	M(=)
Reason Code	M	M(=)
Additional Detail	U	U(=)

Argument

This parameter carries the information associated with the AR-Abort service.

Locally Generated

This parameter specifies if the abort was locally generated, or not.

Originator

This parameter identifies the originator for the abort. Its valid values are DLL, FAL, or FAL-USER. The value DLL cannot be used in the request primitive.

Reason Code

This parameter indicates the reason for the Abort. It may be supplied by either the provider or the user. One reason is defined: AR ASE error. Other reason code values may be supplied by the Data Link Layer, or by the user.

Additional Detail

This optional parameter contains user data that accompanies the indication. When used, the value submitted in the request primitive is delivered unchanged in the indication primitive.

7.2.3.3.5.2 Service Procedure

The abort service is a service that operates through a queue or buffer.

If the user wishes to abort an AR, it submits an abort request primitive to its FAL AR ASE. If an AR ASE detects an unrecoverable local failure or a communication failure, it delivers an abort indication primitive to the endpoint user.

The FAL AR ASE builds an abort request APDU and conveys it on the specified AR if it is capable of doing so. It also transitions the AREP state to CLOSED

Upon receipt of the Abort Request APDU, each remote FAL AR ASE transitions its AREP to CLOSED and delivers an abort indication primitive to its user.

7.2.3.3.6 AR-Compel Service

This service is used by the FAL user to request the AR ASE to convey a message which has been deferred until explicitly released.

NOTE The services that operate on ARs are described abstractly in such a way that they are capable of operating with ARs that convey FAL APDUs through buffers or queues. These services may be implemented such that the capability is provided to load the buffer/queue, and subsequently post it for transfer by the underlying Data Link Layer using this service.

7.2.3.3.6.1 Service Primitives

The service parameters for this service are shown in Table 25.

Table 25 – AR-Compel Service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Schedule ID	U			
Result (+)				S
AREP				M
Status				M
Result (-)				S
AREP				M
Error Info				M

Argument

The argument contains the parameters of the service request.

Schedule ID

This optional parameter specifies the Data Link Layer sequence to be compelled for network scheduled ARs. The scheduling sequence is part of the DL-Mapping defined in IEC 61158-6.

Result(+)

This selection type parameter indicates that the service request succeeded.

Status

This parameter indicates the result of the service request. The following three status codes provided by the DLL are defined:

- success
- failure – inappropriate request;
- failure – reason unspecified.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.3.3.6.2 Service Procedure

The AR-Compel service is a service that operates through a queue or buffer.

The requesting user submits an AR compel.request primitive to its FAL AR ASE. The FAL AR ASE issues the corresponding Data Link Layer service request to the Data Link Layer.

7.2.3.3.7 AR-Get Buffered Message Service

This local service is used by an application process to request the AR ASE to retrieve a message which is being maintained in a buffer in the local Data Link Layer.

This service does not result in the conveyance of an APDU. It is provided so that the FAL user may access a buffer through the FAL AR.

7.2.3.3.7.1 Service Primitives

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

Table 26 – AR-Get Buffered Message Service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Result(+)				S
AREP				M
Decoded buffer data				M
Local timeliness				C
Remote timeliness				C
Duplicate FAL PDU Body				C
Result (-)				S
AREP				M
Error Info				M

Argument

The argument contains the parameters of the service request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Decoded Buffer Data

This parameter contains the user data in the FAL APDU read from the buffer.

Local Timeliness

This conditional parameter, if it is present and True, indicates that the Decoded Buffer Data parameter has met the receiving timeliness criteria defined for the DLL. If its value is False, at least one of the timeliness criteria has not been met. It is present if supported in the DL Mapping Attributes of the AREP.

Remote Timeliness

This conditional parameter, if it is present and True, indicates that the Decoded Buffer Data parameter has met the Publisher's and transmitting DL's timeliness criteria. If its value is False, at least one of the timeliness criteria has not been met. It is present if supported in the DL Mapping Attributes of the AREP.

Duplicate FAL PDU Body

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the Data Link Layer. It is present if supported in the DL Mapping Attributes of the AREP.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.3.3.7.2 Service Procedure

This service requests the FAL to return the current contents of the buffer in a confirmation(+) primitive. If the buffer is empty, a confirmation(-) primitive is returned.

7.2.3.3.8 AR-Schedule Communication Service

This local service provides the AL user with the ability to schedule a sequence of DL-COMPELS for a particular relationship. It maps directly to the DL-Schedule-Sequence service and has no effect on the AR state machines.

This service does not result in the conveyance of an APDU. It may, however, result in the transfer of Data Link Layer PDUs.

7.2.3.3.8.1 Service Primitives

The service parameters for this service are shown in Table 27.

Table 27 – AR-Schedule Communication Service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Invoke ID	M			
Schedule Information	M			
Result (+)				S
AREP				M
Invoke ID				M
Sequence ID				M
Result (-)			S	S
AREP			M	M
Invoke ID			M	M
Error Info			M	M

Argument

The argument contains the parameters of the service request.

Schedule Info

This specifies the Data Link Layer scheduling information for the scheduled communication.

Result(+)

This selection type parameter indicates that the service request succeeded.

Sequence ID

This parameter provides a short-hand identifier for the requested scheduled communication.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.3.3.8.2 Service Procedure

This service requests the communication stack to create a scheduled sequence.

7.2.3.3.9 AR-Cancel Scheduled Sequence Service

This local service provides the AL user with the ability to cancel an existing sequence which has previously been scheduled. It maps directly to the DL-Cancel-Schedule service and has no effect on the AR state machines.

This service does not result in the conveyance of an APDU. It may, however, result in the conveyance of Data Link Layer PDUs.

7.2.3.3.9.1 Service Primitives

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

Table 28 – AR-Cancel Scheduled Sequence Service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Invoke ID	M			
Sequence ID	M			
Result (+)				S
AREP				M
Invoke ID				M
Result (-)				S
AREP				M
Invoke ID				M
Error Info				M

Argument

The argument contains the parameters of the service request.

Sequence ID

This parameter specifies the schedule of the sequence to be cancelled.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.3.3.9.2 Service Procedure

This service requests the communication stack to cancel an existing scheduled sequence.

7.2.3.3.10 AR-Get DL-Time Service

This service is defined for the FAL to provide the FAL User with access to the DL-Time service of the Data Link Layer. To maintain compatibility with the DL-Time service, the service and parameters definitions are not defined in this standard. See IEC 61158-3 for the definition of the DL-Time service.

7.2.3.3.11 AR-Status Service

This local service provides the AL user notification of a status change of the AREP.

7.2.3.3.11.1 Service Primitives

The service parameters for this service are shown in Table 29.

Table 29 – AR-Status

Parameter name	Req	Ind	Rsp	Cnf
Argument		M		
AREP		M		
Status code		M		

Argument

This parameter carries the parameters of the service invocation.

Status Code

This specifies the status change being reported. The following status codes are defined

- lower layer reset;
- buffer received;
- buffer transmitted;
- transmission not timely;
- lower layer lost schedule -- rescheduling required;
- local confirmation.

7.2.3.3.11.2 Service Procedure

This service indicates that a significant event, as defined by the status code parameter, occurred in the communication stack.

7.2.3.3.12 AR-XON-OFF Service

This local service causes the flow of data on a specified AR to be suspended or resumed.

7.2.3.3.12.1 Service Primitives

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

Table 30 – AR-XON-OFF

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Invoke ID	M	M(=)
XON-OFF	M	M(=)

Argument

The argument contains the parameters of the service request.

XON-OFF

This parameter contains user supplied data that can be conveyed with the service request. User data can be "ON" or "OFF".

7.2.3.3.12.2 Service Procedure

The AR-XON-OFF Service is a service that operates through a queue.

The requesting FAL ASE submits an AR-XON-OFF request primitive to its AR ASE. The AR ASE builds an AR-XON-OFF request APDU.

NOTE The transmission schedule is managed by the underlying layer, not the AR-ASE. Refer to IEC 61158-3 and IEC 61158-4 for further details.

Upon receipt of the AR-XON-OFF request APDU, the receiving AR ASE delivers an AR-XON-OFF.indication primitive to the appropriate FAL ASE as indicated by the FAL Service Type Parameter.

7.2.3.3.13 AR Remote Read Service

This service provides the application process to request the AR ASE to retrieve a message which is being maintained in a buffer in the remote Data Link Layer.

7.2.3.3.13.1 Service Primitives

The service parameters for this service are shown in Table 31.

Table 31 – AR-Remote Read Service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Priority	M			
Result (+)				S
AREP				M
Decoded Buffer Data				M
Local Timeliness				C
Remote Timeliness				C
Result (-)				S
AREP				M
Error info				M

Argument

The argument contains the parameters of the service request.

Priority

This argument is locally used and defined by the user to allow two data flows

Result(+)

This selection type parameter indicates that the service request succeeded.

Decoded Buffer Data

This parameter contains the user data in the FAL APDU read from the buffer.

Local timeliness

This conditional parameter, if it is present and True, indicates that the Decoded Buffer Data parameter has met the receiving timeliness criteria defined for the DLL. If its value is False, at least one of the timeliness criteria has not been met. It is present if supported in the DL Mapping Attributes of the AREP.

Remote timeliness

This conditional parameter, if it is present and True, indicates that the Decoded Buffer Data parameter has met the Publisher's and transmitting DL's timeliness criteria. If its value is False, at least one of the timeliness criteria has not been met. It is present if supported in the the DL Mapping Attributes of the AREP.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.3.3.13.2 Service Procedure

This service requests the FAL to return the current contents of the remote buffer and give the value of the contents in a confirmation(+) primitive. If the buffer is empty, a confirmation(-) primitive is returned.

7.2.3.3.14 AR-Remote Write Service

This service provides the AL user with the ability to trigger a data exchange, after writing a value in a local buffer.

7.2.3.3.14.1 Service Primitives

The service parameters for this service are shown in Table 32.

Table 32 – AR-Remote write service

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Priority	M			
Decoded Buffer Data	M			
Result (+)				S
AREP				M
Result (-)				S
AREP				M
Error Info				M

Argument

The argument contains the parameters of the service request.

Priority

This argument is locally used and defined by the user to allow two data flows.

Decoded Buffer Data

This argument is used to contain the APDU to be written.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.3.4 Service Procedure

This service requests the FAL to write the contents of the local buffer and to trigger the data transfer. A confirmation(+) primitive is sent when the service succeeds. If the service fails, a confirmation(-) primitive is returned.

7.2.4 Variable ASE

7.2.4.1 Overview

In the Fieldbus environment, application processes contain data that remote applications are able to read and write. The variable ASE defines the network visible attributes of application data and provides a set of services used to read, write, and report their values. Common FAL management services are used to create and delete variable objects and to access their attributes.

Two classes of variable APOs are defined by the Variable Model, individual variables and variable lists. Individual variables are used to specify access to application data of any FAL-defined or user-defined data type. Services defined to access individual variables can be used to access one or more entire variables or one or more entries (per variable) in an array or one or more fields (per variable) in a structure.

Variable lists are used to group variables for access purposes. The same services used to access individual variables can be used to access a variable list.

Two types of operation are supported by variable services, "best effort" and "atomic". The "best effort" services succeed when at least one of the referenced values can be accessed. The "atomic" services succeed when all of the referenced values can be accessed, and fail if any one of the values cannot be accessed.

When supported by the appropriate type of application relationship, the Variable Model service can be used to support two different access models, the client/server model and the publisher/subscriber 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 server's activity is stimulated by clients on the network; if there are no requests, the server generates no responses.

The publisher/subscriber model is different. It is characterized by a data producer publishing its data onto the network. Subscribers wishing to acquire the published data join the application relationship used to publish it and listen for the data as it is transmitted.

Two models are provided to support this publisher/subscriber activity, the "pull" and the "push". In the "pull" model, the publishing manager pulls the data from the publisher by issuing a read request to it. The publisher responds by multicasting a sequence of read responses to the publishing manager and to the subscribers.

In the "push" model, the publishing manager is always co-located with the publisher. Subscribers are able to indicate to the publishing manager how they want the data published. Through local interfaces, the publishing manager controls the activity of the publisher and the characteristics of the application relationships used to distribute the data. In this case, the published data is transmitted onto the network using the Information Report unconfirmed service.

Subscribers in both models receive the published data through a local copy of the data maintained by the FAL. As the data is received from the network, the local copy is updated and made available to the subscriber. When the subscriber wishes to access the data, it accesses the local copy, instead of issuing a read request to the remote variable as would happen in the client/server model. If the appropriate AREP attributes are set, timeliness information will be included with the data.

The FAL AR ASE supports both publisher/subscriber models by multi-casting the published data on buffered, network scheduled application relationships. The exact characteristics of the transfers are controlled by the attribute settings of the application relationship.

The formal model of the variable model is presented next, followed by a description of its services. IEC 61158-6 describes the abstract syntax and procedures for its protocol.

7.2.4.1.1 Requirements for Access to Variables

The structure of a variable value is defined by its data type. A variable's data type may be of any valid standardized or user defined data type. Standardized types and the facilities for defining user types are specified by the Data Type Model.

As the Data Type Model permits nesting of data structures and arrays to more than one level, the services defined for variable access permit partial access to any nesting level. Partial access means independent access to several components of each structure or array referenced. That is, if constructed type "A" contains constructed type "B", then partial access is provided to "B", and also to components of "B".

7.2.4.2 Variable Model Class Specification

7.2.4.2.1 Simple Variable Formal Model

FAL ASE:	VARIABLE ASE
CLASS:	SIMPLE VARIABLE
CLASS ID:	7
PARENT CLASS:	TOP

ATTRIBUTES:

1	(o)	Key Attribute:	Symbolic Address
2	(m)	Attribute:	Data Type
3	(m)	Attribute:	Length
4	(c)	Constraint:	Data Type Format = STRING
4.1	(o)	Attribute:	Variable Length Conveyance (TRUE, FALSE) -- see note below
5	(m)	Attribute:	Access Privilege
5.1	(m)	Attribute:	Password
5.2	(m)	Attribute:	Access Groups
5.3	(m)	Attribute:	Access Rights
6	(m)	Attribute:	Local Detail

SERVICES:

1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
3	(o)	OpsService:	Read List
4	(o)	OpsService:	Write List
5	(o)	OpsService:	Information Report
6	(o)	OpsService:	Information Report List
7	(o)	OpsService:	Exchange
8	(o)	OpsService:	Exchange List

NOTE The constraint is TRUE when a variable is being defined as an array.

7.2.4.2.1.1 Attributes

Symbolic Address

This optional key attribute specifies a symbolic reference for the variable. This attribute provides a second name space for defining variables that is separate from that of the variable name to ensure that duplication of names in separate name spaces do not create a problem. The symbolic address may be used to assign variable names independent of how they are named within a given system.

A space (" ") character may not be embedded in the middle of a symbolic address. However, using it as leading or trailing fillers is permitted. Such fillers are not to be interpreted as part of the symbolic address. The use of leading or trailing fillers permits fixed-length character string usage.

NOTE For example, a device manufacturer may use the symbolic address to name the data acquired from a local input channel as "port 1". A user of that data may create a second variable for the data using the variable name "temperature".

Data Type

This attribute is the numeric identifier of a FIXED-LENGTH or STRING data type.

Length

This attribute indicates the length of simple variables data types. For variables with STRING data types, this attribute is used to define the maximum length of the variable in octets. For variables with the FIXED-LENGTH data type, this attribute is used to reflect the length of the variable as specified by the data type.

Variable Length Conveyance

This conditional Boolean attribute, when TRUE, indicates that only the valid octets of the string are conveyed. When FALSE, all octets of the string are conveyed, even if they are not part of the string value. This attribute is present only for variables with the STRING data type.

Access Privilege

This attribute specifies the access controls defined for this variable. It is composed of the following:

Password

This attribute contains the password for the access rights. Its value is null if it is not used.

Access Groups

This attribute identifies which of the eight user-defined access groups are defined for the variable. More than one access group may be defined.

Access Rights

This attribute defines the type of access defined for the variable. Valid values are:

- Right to Write for Access Groups
- Right to Read for Access Groups
- Right to Write for the registered Password
- Right to Read for the registered Password
- Right to Write for all Communication Partners
- Right to Read for all Communication Partners

Local Detail

This attribute contains local information.

7.2.4.2.1.2 Services

All services defined for this class are optional. When an instance of the class is defined, at least one has to be supported.

Read

This optional service may be used to read a single variable object or variable list object. This service may be used in both the client/server model and the publisher/subscriber pull model.

Write

This optional service may be used to update a single variable object or variable list object. This service may be used only in the client/server model.

Read List

This optional service may be used to read multiple variable objects or components of multiple variable objects or a single variable list object. The number of referenced variables/components or variable objects in the list is a device constructor matter. Each of the referenced variables/components or variables in the list is accessed and returned in a “best effort” fashion, such that at least one value is to be returned for the service to succeed. This service may be used only in the client/server model.

Write List

This optional service may be used to update multiple variable objects or single components of multiple variable objects or a single variable list object. The number of referenced variables/components or variable objects in the list is a device constructor matter. Each of the referenced variables/components or variables in the list are updated in a “best effort” fashion, such that at least one value is to be updated for the service to succeed. This service may be used only in the client/server model.

Information Report

This optional service is an unconfirmed service that may be used to report the value of a variable or variable list object. This service may be used in both the client/server model and the publisher/subscriber push model.

Information Report List

This optional unconfirmed service may be used to report multiple variable objects or single components of multiple variable objects or a single variable list object. The number of referenced variables/components or variable objects in the list is a device constructor matter. This service may be used in both the client/server model and the push publisher/subscriber model.

Exchange

This optional service is a confirmed service that may be used to write the value of a remote variable or variable list and read the value of another variable or variable list in one operation. This service may be used in the client/server model. The relationship between the specified variables and User Layer Exchange Function is outside the scope of this standard.

Exchange List

This optional confirmed service may be used to update multiple variable objects or single components of multiple variable objects or a single variable list object and to read multiple variable objects or single components of multiple variable objects or a single variable list object in one operation. The number of referenced variables/components or variable objects in the list is a device constructor matter. Each of the referenced variables/components or variables in the list is updated/accessed in a “best effort” fashion, such that at least one value is to be updated/returned for the service to succeed. This service may be used only in the client/server model. The relationship between the specified variables and a User Layer Exchange Function is outside the scope of this standard.

7.2.4.2.2 Array Variable Formal Model

FAL ASE:	VARIABLE ASE
CLASS:	ARRAY VARIABLE
CLASS ID:	8
PARENT CLASS:	TOP

ATTRIBUTES:

1	(o)	Key Attribute:	Symbolic Address
2	(m)	Attribute:	Data Type
2.1	(s)	Attribute:	Data Type ID
2.2	(s)	Attribute:	Embedded Data Type
3	(c)	Constraint:	Data Type Format = ARRAY FIXED-LENGTH STRING
3.1	(o)	Attribute:	Element Length
4	(c)	Constraint:	Data Type Format = STRING
4.1	(o)	Attribute:	Variable Length Conveyance (TRUE, FALSE)
5	(o)	Attribute:	Number of Elements
6	(m)	Attribute:	Access Privilege
6.1	(m)	Attribute:	Password
6.2	(m)	Attribute:	Access Groups
6.3	(m)	Attribute:	Access Rights
7	(m)	Attribute:	Local Detail

SERVICES:

1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
3	(o)	OpsService:	Read List
4	(o)	OpsService:	Write List
5	(o)	OpsService:	Information Report
6	(o)	OpsService:	Information Report List
7	(o)	OpsService:	Exchange
8	(o)	OpsService:	Exchange List

7.2.4.2.2.1 Attributes

Symbolic Address

See the description of this attribute defined for the Simple Variable class above.

Data Type

This attribute specifies the data type for elements of the array.

Data Type ID

This attribute is the numeric identifier of a data type associated with the array. If the format of the data type is ARRAY, this attribute is the data type of the array. If the data type is FIXED-LENGTH or STRING, this attribute is the data type of an array element.

Embedded Data Type

This attribute is used to embed the data type description in the array definition. The contents for this attribute are shown in the Data Type model.

Element Length

This conditional attribute indicates the length of an array element with an ARRAY, FIXED-LENGTH or STRING data type. For array elements with STRING data types, this attribute is used to define the length of an array element in octets. For variables with an ARRAY or FIXED-LENGTH data type, this attribute is used to reflect the length of an array element as specified by the data type.

Variable-length Conveyance

This conditional Boolean attribute, when TRUE, indicates that only the valid octets of the string are conveyed. When FALSE, all octets of the string are conveyed, even if they are not part of the string value. This attribute is present only for arrays with a STRING data type.

Number of Elements

This attribute specifies the number of array elements for variable objects that are defined as arrays. Array variables may be defined in one of two ways. Each of the ways, and the use of this attribute for each are shown below.

Method

Reference to a data type with the format of ARRAY:

Reference to a data type with the format of FIXED-LENGTH or STRING

Attribute Use

Reflects the number of elements defined for the array data type

Specifies the number of elements defined for the array variable

Access Privilege

See the description of this attribute and its components defined in the Simple Variable class above.

Local Detail

This attribute contains local information.

7.2.4.2.2.2 Services

See the description of the services defined in the Simple Variable class above.

7.2.4.2.3 Record Variable Formal Model

FAL ASE:	VARIABLE ASE
CLASS:	RECORD VARIABLE
CLASS ID:	9
PARENT CLASS:	TOP

ATTRIBUTES:

1	(o)	Key Attribute:	Symbolic Address
2	(m)	Attribute:	Data Type
2.1	(s)	Attribute:	Data Type ID
2.2	(s)	Attribute:	Embedded Data Type
3	(m)	Attribute:	Access Privilege
3.1	(m)	Attribute:	Password
3.2	(m)	Attribute:	Access Groups
3.3	(m)	Attribute:	Access Rights
4	(m)	Attribute:	List of Local Detail

SERVICES:

1	(o)	OpsService:	Read
2	(o)	OpsService:	Write
3	(o)	OpsService:	Read List
4	(o)	OpsService:	Write List
5	(o)	OpsService:	Information Report
6	(o)	OpsService:	Information Report List
7	(o)	OpsService:	Exchange
8	(o)	OpsService:	Exchange List

7.2.4.2.3.1 Attributes

Symbolic Address

See the description of this attribute defined for the Simple Variable class above.

Data Type

This attribute specifies the data type for elements of the array.

Data Type ID

This attribute is the numeric identifier of a STRUCTURE data type.

Embedded Data Type

This attribute is used to embed the data type description in the record definition. The contents for this attribute are shown in the Data Type model.

Access Privilege

See the description of this attribute defined for the Simple Variable class above.

Password

See the description of this attribute defined for the Simple Variable class above.

Access Groups

See the description of this attribute defined for the Simple Variable class above.

Access Rights

See the description of this attribute defined for the Simple Variable class above.

List of Local Detail

This attribute contains the local detail for each element (field) of the record.

7.2.4.2.3.2 Services

See the description of the services defined for the Simple Variable class above.

7.2.4.2.4 Variable List Formal Model

FAL ASE:		VARIABLE ASE
CLASS:		VARIABLE LIST
CLASS ID:		10
PARENT CLASS:		TOP
ATTRIBUTES:		
1	(m)	Attribute: Number of Entries
2	(m)	Attribute: List Of Variables
3	(c)	Attribute: Deletable
4	(m)	Attribute: Access Privilege
4.1	(m)	Attribute: Password
4.2	(m)	Attribute: Access Groups
4.3	(m)	Attribute: Access Rights
SERVICES:		
1	(o)	OpsService: Read
2	(o)	OpsService: Write
3	(o)	OpsService: Read List
4	(o)	OpsService: Write List
5	(o)	OpsService: Exchange
6	(o)	OpsService: Exchange List
7	(o)	OpsService: Information Report
8	(o)	OpsService: Information Report List

7.2.4.2.4.1 Attributes

Number of Entries

This attribute contains the number of variables in the list.

List of Variables

This attribute identifies the variables (only simple, array or record variable objects) by Key Attribute that are contained in the list.

Deletable

This attribute indicates, when TRUE, that the variable list can be deleted using the Delete service. The value of this attribute is always TRUE for objects dynamically created with the Object Management ASE Create Service.

Access Privilege

This attribute specifies the access controls defined for this variable list. It is composed of the following:

Password

This attribute contains the password for the access rights. Its value is null if it is not used.

Access Groups

This attribute identifies which of the eight user-defined access groups are defined for the variable list. More than one access group may be defined.

Access Rights

This attribute defines the type of access defined for the variable list. Valid values are:

- Right to Delete for Access Groups
- Right to Write for Access Groups
- Right to Read for Access Groups
- Right to Delete for the registered Password
- Right to Write for the registered Password
- Right to Read for the registered Password
- Right to Delete for all Communication Partners
- Right to Write for all Communication Partners
- Right to Read for all Communication Partners

7.2.4.2.4.2 Services

See the description of the services defined for the Simple Variable class above.

7.2.4.3 Variable ASE Service Specification

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

Read
Write
Read List
Write List
Information Report
Information Report List
Exchange
Exchange List

The exact nature of the operation of variable services is determined, in part, by the application relationship over which they operate.

7.2.4.3.1 Read Service

This confirmed service may be used to read the value of a variable object or variable list object. It may be used with application relationships configured to support the client/server model, or it may be used with application relationships configured to support the publisher/subscriber "pull" model.

7.2.4.3.1.1 Service Primitives

The service parameters for this service are shown in Table 33.

Table 33 – Read service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Variable Specifier	M	M(=)		
Key Attribute	S	S(=)		
Key Attribute and Component ID	S	S(=)		
Numeric Address and Data Length	S	S(=)		
Data Type Requested	U	U(=)		
Object Revision Requested	U	U(=)		
Best Effort Requested	U	U(+)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
List of Access Results			M	M(=)
Error Status			S	S(=)
Returned Data			S	S(=)
Value			M	M(=)
Data Type			C	C(=)
Object Revision			C	C(=)
Timeliness			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

This parameter carries the parameters of the service invocation.

Variable Specifier

This selector type parameter specifies the variable, variable list, or an element of an array or record variable.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Key Attribute and Component Identifier

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify a component with a nesting level greater than one if supported by the AP.

Numeric Address/Data Length

This parameter provides the numeric address and the data length of the data to be read. The mapping between this parameter and the actual memory address in a real system is a local matter.

Data Type Requested

This optional parameter indicates that the data type of each variable should be returned with the data.

Object Revision Requested

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

Best Effort Requested

This conditional, optional parameter is used to request to perform a “best effort” read of a variable list. In the “best effort” read, the service succeeds if at least one variable in the variable list can be read.

Result(+)

This selection type parameter indicates that the service request succeeded.

List of Access Result

This parameter contains the responses returned by the remote AP. If “best effort” was requested for a variable list, then a list of responses is returned. The order of the response in the list is the same as the order of the variables in the variable list.

Error Status

This selection type parameter is when “best effort” was requested for a variable list and one of the variables in the list could not be read. It indicates the reason for the read failure.

Returned Data

This selection type parameter contains the data returned by the remote AP. It is always present if “best effort” was not requested for a variable list.

Value

This parameter contains the value read. For each of the variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated together in the order that they appear in the list. If any of the variables in a variable list could not be read, the service fails.

Data Type

This optional parameter indicates that the data type of each value is returned.

Object Revision

This conditional parameter contains the object revision of the specified object. It is present if the object revision was requested.

Timeliness

This conditional parameter indicates the Data Link Layer timeliness status for the referenced variable, or variable list object. This parameter is present if it is supported on the specified AR.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.4.3.1.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Read Service is an "all or nothing" service. All or nothing means that the service succeeds only if the requested value, or values in the case of a variable list object, are read and returned.

If the object specified to read is a variable list object, the value, data type (when requested), and object revision (when requested) of the variables in the variable list are returned.

A timeliness parameter is included in the indication primitive if the AR that conveyed the APDU Body supports timeliness.

7.2.4.3.2 Read List Service

This confirmed service is used to read the values of multiple variables or a single variable list object. It may be used with application relationships configured to support the client/server model.

It operates in a "best effort" fashion, such that the service succeeds if the value for at least one variable or one variable in a variable list is returned, otherwise it fails. When the variable was successfully accessed, the returned data must be the variable value otherwise it must be an error status.

7.2.4.3.2.1 Service Primitives

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

Table 34 – Read List Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Data Type Requested	U	U(=)		
List Of Variable Specifiers	S	S(=)		
Key Attribute	S	S(=)		
Key Attribute and Component ID(s)	S	S(=)		
List Of Numeric Addresses and Data Lengths	S	S(=)		
Object Revision Requested	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
List of Access Results			M	M(=)
List of Data Types			C	C(=)
List of Data			M	M(=)
Error Status			S	S(=)
Value			S	S(=)
Value with Object Revision			S	S(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

This parameter carries the parameters of the service invocation.

Data Type Requested

This optional parameter is used to request that the type description of the List of data parameter be returned. A value of TRUE indicates that the type description is desired. A value of FALSE indicates that type description is not desired. This request can be used only with List of Variable Specifiers selection.

List of Variable Specifiers

This parameter individually identifies a list of variables and/or components of variables to be read or one variable list to be read. The nesting level of the components to be read can be equal to or greater than one.

Key Attribute

This selection type parameter specifies a Variable or Variable list Identifier key attribute values.

Key Attribute And Component ID(s)

This parameter identifies the components of a constructed data type variable by a combination of one of its key attributes and the components path ID(s).

List Of Numeric Addresses and Data Lengths

This parameter provides the numeric address and the data length of the variable objects to be read. The mapping between this parameter and the actual memory address in a real system is a local matter. This selection is allowed to access only the data of variable object

NOTE The Numeric Address parameters may be useful to specify a variable or a certain memory location in a called service user. These addresses are not global. One example of the use of the parameters is to specify a pointer to a target memory location.

Object Revision Requested

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired for all variables (if supported). A value of FALSE indicates that object revision is not desired for any variable.

Result(+)

This selection type parameter indicates that the service request succeeded.

List of Access Results

This parameter contains an indicator for each variable accessed. If an access succeeds the indicator must be TRUE otherwise it must be FALSE.

List of Data Types

This optional parameter provides the type description of the values in the List of Data parameter.

List of Data

This parameter contains one or more error statuses, variable values, or variable values with object revision read. The error statuses or values with/without object revision are concatenated in the order in which they were described in the request. For variable lists this is always the order of the list definition. For each variable or a variable in a variable list successfully accessed, the returned data must be the value or the value with object revision of the variable object (if requested). Otherwise it must be an error status. The returned data must also be an error status if the object revision is requested and the variable object accessed does not support the object revision attribute.

Error Status

This parameter indicates the most probable reason why the operation failed using the error class and error code defined for the read service.

Value

This parameter contains a value read. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Value with Object Revision

This parameter contains a value read plus the object revision of the variable object. For variables, this parameter contains the value of the variable. For variable lists, this parameter

contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.4.3.2.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Read List Service is a "best effort". Best effort means that the service succeeds if at least one value is read.

If the user is not able to read at least one of the values, the service fails and the user issues a Read List Service response (-) primitive indicating the reason.

When requested, the data type descriptions are returned. When requested, the object revision attribute value is returned concatenated with the value of the variable.

7.2.4.3.3 Write Service

This confirmed service is used to write the value of a variable. It is not used with the publisher/subscriber model.

7.2.4.3.3.1 Service Primitives

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

Table 35 – Write Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)	M(=)	M(=)
AREP	M	M(=)		
Invoke ID	M	M(=)		
Variable Specifier	M	M(=)		
Key Attribute	S	S(=)		
Key Attribute and Component ID	S	S(=)		
Numeric Address and Data Length	S	S(=)		
Value	M	M(=)		
Data Type	U	U(=)		
Object Revision	U	U(=)		
Best Effort Requested	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
List of Error Status			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Variable Specifier

This parameter specifies the variable, variable list, or an element of an array or record variable.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Key Attribute and Component Identifier

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify a component with a nesting level greater than one if supported by the AP.

Numeric Address/Data Length

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

Value

This parameter contains the value to write. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated together in the order that they appear in the list.

NOTE If any variable in a variable list object cannot be updated, none of the variables in the variable list object will be updated, and the write will fail.

Data Type

This optional parameter contains the data type of the values in the Value parameter.

Object Revision

This optional parameter contains the expected object revision of the variable object. When present, it indicates that the AP containing the variable is to compare the value of this parameter to the Object Revision of the variable before applying the write.

Best Effort Requested

This conditional, optional parameter is used to request to perform a "best effort" write of a variable list. In the "best effort" write, the service succeeds if at least one variable in the variable list can be written.

Result(+)

This selection type parameter indicates that the service request succeeded.

List of Status

This parameter contains more than one status if a "best effort" write was requested. The status indicates success or the reason for failure. The order in the list is the same as the order of the variables in the variable list.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.4.3.3.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Write Service is an "all or nothing" service. All or nothing means that the service succeeds only if the requested value, or values in the case of a variable list object, are successfully written.

7.2.4.3.4 Write List Service

This confirmed service is used to write the values of multiple variables or variables in a single variable list object. The number of variables referenced or defined in the list is a device local matter. It may be used with application relationships configured to support the client/server model.

It operates in a "best effort" fashion, such that the service succeeds if the value for at least one variable or variable in a variable list is written, otherwise it fails.

7.2.4.3.4.1 Service Primitives

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

Table 36 – Write List Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
List of Variable Specifiers	S	S(=)		
Key Attribute	S	S(=)		
Key Attribute and Component ID(s)	S	S(=)		
List of Numeric Addresses and Data Lengths	S	S(=)		
List of Data Types	U	U(=)		
List of Data	M	M(=)		
Value	S	S(=)		
Value with Object Revision	S	S(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
List of Variable Data Access Statuses			M	M(=)
List of Error Statuses			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

This parameter carries the parameters of the service invocation.

List of Variable Specifiers

This parameter individually identifies multiple variables and/or components of variables to be written or a single variable list to be written. The nesting level of the components to be written can be equal to or greater than one.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Key Attribute and Component ID(s)

This parameter identifies components of a constructed data type variable by a combination of one of its key attributes and the component path ID(s).

List of Numeric Addresses and Data Lengths

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

List of Data Types

This optional parameter provides the type description of the values in the List of Data parameter.

List of Data

This parameter contains one or more values or one or more values with object revision to write. The values with/without object revision are concatenated in the order in which they are described in the preceding parameters. For variable lists this is always the order of the list definition.

Value

This parameter contains a value to write. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Value with Object Revision

This parameter contains a value to write plus the expected object revision of the variable object. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Result(+)

This selection type parameter indicates that the service request succeeded.

List of Variable Data Access Statuses

This parameter contains an indicator for each variable accessed. If an access succeeds, the indicator must be TRUE, otherwise it must be FALSE.

List of Error Statuses

This parameter contains an error status for each individual variable or variable in variable list for which the access failed. The order of the error statuses is the same as the order of the objects identified in the service request.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.4.3.4.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Write List Service is a "best effort" service. Best effort means that the service succeeds if at least one value is written.

If the responding user is able to write the value for at least one of the requested variables, the user returns a Write List Service response (+) primitive.

If the user is not able to write at least one of the values, the user issues a Write List Service response (-) primitive indicating the reason.

7.2.4.3.5 Information Report Service

This unconfirmed service is used by an application process to report the value of a variable to receiver(s) designated by the AR.

7.2.4.3.5.1 Service Primitives

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

Table 37 – Information Report Service

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Destination DL-Address	C	
Source DL-Address		C
Variable Specifier	C	C(=)
Key Attribute	S	S(=)
Key Attribute and Component ID	S	S(=)
Numeric Address and Data Length	S	S(=)
Value	M	M(=)
Object Revision	U	U(=)
Data Type	U	U(=)
Timeliness		C
Duplicate FAL PDU Body		C

Argument

This parameter carries the parameters of the service invocation.

Destination DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It gives the remote address to which the requested Information Report should be sent.

Source DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It identifies the source address from which the Information Report is to be sent.

Variable Specifier

This selector type parameter specifies the variable, variable list, or an element of an array or record variable.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Key Attribute and Component Identifier

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify a component with a nesting level greater than one if supported by the AP.

Numeric Address and Data Length

This parameter provides the numeric address and data length reported.

Value

This parameter contains the value. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated together in the order that they appear in the list.

Object Revision

This optional parameter contains the expected object revision of the variable object. When present, it indicates that the calling service user wishes the server to check the current Object Revision parameter of the variable to be updated. If supported, the server only updates the value of a variable in the list if its object revision attribute is equivalent to the value of this parameter.

Data Type

This optional parameter contains the data type of the values in the Value parameter.

Timeliness

This conditional parameter indicates the Data Link Layer timeliness status for the referenced variable, or variable list object. It is present if timeliness is supported on the specified AR.

Duplicate FAL PDU Body

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the Data Link Layer. It is present if supported in the DL Mapping Attributes of the AREP.

7.2.4.3.5.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

A timeliness parameter is included in the indication primitive if the AR that conveyed the APDU Body supports timeliness.

7.2.4.3.6 Information Report List Service

This unconfirmed service is used by an application process to report a list of values or a list of values with object revision to receiver(s) designated by the AR. The number of variables referenced or defined in the list is a device local matter.

7.2.4.3.6.1 Service Primitives

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

Table 38 – Information Report List Service

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Destination DL-Address	C	
Source DL-Address		C
List of Variable Specifiers	S	S(=)
Key Attribute	S	S(=)
Key Attribute and Component IDs	S	S(=)
List of Numeric Addresses and Data Lengths	S	S(=)
List of Data Types	U	U(=)
List of Data	M	M(=)
Value	S	S(=)
Value with Object Revision	S	S(=)
Timeliness		C
Duplicate FAL PDU Body		C

Argument

This parameter carries the parameters of the service invocation.

Destination DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It gives the remote address to which the requested Information Report List is to be sent.

Source DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the source address from which the indicated Information Report List is to be sent.

List of Variable Specifiers

This parameter individually identifies multiple variables and/or components of variables to be written or a single variable list to be written. The nesting level of the components to be written can be equal to or greater than one.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Key Attribute and Component ID(s)

This parameter identifies components of a constructed data type variable by a combination of one of its key attributes and the components path ID(s).

List of Numeric Addresses and Data Lengths

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

List of Data Types

This optional parameter provides the type description of the values in the List of Data parameter.

List of Data

This parameter contains one or more values or one or more values with object revision to write. The values with/without object revision are concatenated in the order in which they are described in the preceding parameters. For variable lists this is always the order of the list definition.

Value

This parameter contains a value to write. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Value with Object Revision

This parameter contains a value to write plus the expected object revision of the variable object. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Timeliness

This conditional parameter indicates the Data Link Layer timeliness for the list of values. It is present if timeliness is defined for the specified AR.

Duplicate FAL PDU Body

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the Data Link Layer. It is present if supported in the DL Mapping Attributes of the AREP.

7.2.4.3.6.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

If the object specified to be reported is a variable list object, the values or the values with object revision of the variables in the variable list are concatenated in the List of Data parameter in the order that the variables appear in the variable list.

A timeliness parameter is included in the indication primitive if the AR that conveyed the APDU Body supports timeliness.

7.2.4.3.7 Exchange Service

This confirmed service is used to write the value of one variable or variable list object, and read the value of another variable or a variable list object in a single operation. The order of processing the read and write by the responding user is not specified by the FAL.

7.2.4.3.7.1 Service Primitives

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

Table 39 – Exchange Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Variable to Write	M	M(=)		
Specifier	S	S(=)		
Key Attribute	S	S(=)		
Key Attribute and Element ID	S	S(=)		
Numeric Address and Data Length	S	S(=)		
Value	M	M(=)		
Data Type	U	U(=)		
Object Revision	U	U(=)		
Variable to Read	M	M(=)		
Specifier	S	S(=)		
Key Attribute	S	S(=)		
Key Attribute and ElementID	S	S(=)		
Numeric Address and Data Length	S	S(=)		
Data Type Requested	U	U(=)		
Object Revision Requested	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Write Result			M	M(=)
Read Response			M	M(=)
Error Status			S	S(=)
Returned Value			S	S(=)
Value			M	M(=)
Data type			C	C(=)
Object Revision			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Specifier for Variable to Write

This parameter specifies the variable, variable list, or variable field to be updated.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Key Attribute and Element Identifier

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify an element with a nesting level greater than one if supported by the AP.

Numeric Address/Data Length

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

Value to Write

This parameter contains the value to write. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated together in the order that they appear in the list.

NOTE If any variable in a variable list cannot be updated, none of the variables in the variable list object will be updated, and the write will fail.

Data Type

This optional parameter contains the data type of the values to write in the Value parameter.

Object Revision

This optional parameter contains the expected object revision of the variable object to be updated. When present, it indicates that the AP containing the variable is to compare the value of this parameter to the Object Revision of the variable before applying the write.

Variable Specifier

This selector type parameter specifies the variable, variable list, or variable field to be read.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Key Attribute and Element Identifier

This parameter identifies the field of a constructed data type variable by a combination of one of its key attributes and the field identifier of a data structure or an index into an array. This parameter may identify an element with a nesting level greater than one if supported by the AP.

Numeric Address/Data Length

This parameter provides the numeric address and the data length of the data to be read. The mapping between this parameter and the actual memory address in a real system is a local matter.

Data Type Requested

This optional parameter indicates that the data type of each variable should be returned with the data.

Object Revision Requested

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

Result (+)

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

Write Status

This parameter indicates whether or not the variable was updated as requested. If it was not updated, it indicates the most probable reason why the operation failed using the error class and error code defined for the write service.

Read Response

This parameter provides the data value corresponding to the Variable Specifier to Read parameter in the request primitive. This parameter indicates the result of the read operation.

Error Status

This parameter is used to indicate that the access to the specified object was not successful. It indicates the most probable reason why the operation failed using the error class and error code defined for the read service.

Returned Value

This parameter contains the value of the referenced variable or variable list object. If the identified variable was a variable list object, then this parameter will contain a concatenated set of values for each variable in the list.

Value

This parameter contains the value read. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated together in the order that they appear in the list.

NOTE Because there are no identifiers in a concatenated list of values for a variable list object, a read failure for any variable in the variable list object causes the service to fail.

Data Type

This optional parameter indicates that the data type of each value returned.

Object Revision

This conditional parameter contains the object revision of the specified object. It is present if the object revision was requested.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.4.3.7.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Exchange Service is a "best effort" service. Best effort means that the service succeeds if either of the requested read or write operations succeeds.

If a variable list was specified for the variable to write, the value and object revision (when present) of each variable in the variable list are concatenated in the Exchange Request APDU Body in the order that the variables appear in the variable list.

If a variable list was specified for the variable to read, the value and object revision (when present) of each variable in the variable list are concatenated in the Exchange Response APDU Body in the order that the variables appear in the variable list.

7.2.4.3.8 Exchange List Service

This confirmed service is used to write multiple variable objects or single components of multiple variable objects or a single variable list object and to read multiple variable objects or single components of multiple variable objects or a single variable list object in a single operation. The order of processing the read and write by the responding user is not specified by the FAL.

7.2.4.3.8.1 Service Primitives

The service parameters for this service are shown in Table 40.

Table 40 – Exchange List Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Variables to Write	M	M(=)		
List of Variable Specifiers	S	S(=)		
Key Attribute	S	S(=)		
Key Attribute and Component IDs	S	S(=)		
List of Numeric Addresses and Data Lengths	S	S(=)		
List of Data Types	U	U(=)		
List of Data	M	M(=)		
Value	S	S(=)		
Value with Object Revision	S	S(=)		
Variables to Read	M	M(=)		
List of Variable Specifiers	S	S(=)		
Key Attribute	S	S(=)		
Key Attribute and Component IDs	S	S(=)		
List of Numeric Addresses and Data Lengths	S	S(=)		
Data Type Requested	U	U(=)		
Object Revision Requested	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
List of Write Access Results			M	M(=)
List of Error Statuses			C	C(=)
List of Read Access Results			M	M(=)
List of Data Types			C	C(=)
List of Data			M	M(=)
Error Status			S	S(=)
Value			S	S(=)
Value With Object Revision			S	S(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Variables to Write

This parameter contains information related to the variables to be updated.

List of Variable Specifiers

This parameter individually identifies multiple variables and/or components of variables to be written or a single variable list to be written. The nesting level of the components to be written can be equal to or greater than one.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Key Attribute and Component ID(s)

This parameter identifies the component of a constructed data type variable by a combination of one of its key attributes and the components path ID(s).

List of Numeric Addresses and Data Lengths

This parameter provides the numeric address and the data length of the data to be written. The mapping between this parameter and the actual memory address in a real system is a local matter.

List of Data Types

This optional parameter provides the type description of the values in the List of Data parameter.

List of Data

This parameter contains one or more values or one or more values with object revision to write. The values with/without object revision are concatenated in the order in which they are described in the preceding parameters. For variable lists this is always the order of the list definition.

Value

This parameter contains a value to write. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Value with Object Revision

This parameter contains a value to write plus the expected object revision of the variable object. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Variables to Read

This parameter carries information related to the variables to be read.

List of Variable Specifiers

This parameter individually identifies multiple variables and/or components of variables to be written or a single variable list to be read. The nesting level of the components to be read can be equal to or greater than one.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Key Attribute and Component ID(s)

This parameter identifies the component of a constructed data type variable by a combination of one of its key attributes and the component path ID(s).

List of Numeric Addresses and Data Lengths

This parameter provides the numeric address and the data length of the data to be read. The mapping between this parameter and the actual memory address in a real system is a local matter.

Data Type Requested

This optional parameter is used to request that the data type be returned. A value of TRUE indicates that data type is desired. A value of FALSE indicates that data type is not desired.

Object Revision Requested

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired.

Result (+)

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

List of Write Access Results

This parameter contains an indicator for each variable accessed. If an access succeeds the indicator must be TRUE otherwise it must be FALSE.

List of Error Statuses

This parameter contains an error status for each individual variable or variable in variable list for which the access failed. The order of the error statuses is the same as the order of the objects identified in the service request.

List of Read Access Results

This parameter contains an indicator for each variable accessed. If an access succeeds the indicator must be TRUE otherwise it must be FALSE.

List of Data Types

This optional parameter provides the type description of the values in the List of Data parameter.

List of Data

This parameter contains one or more error statuses, values or values with object revision read. The error statuses or values with/without object revision are concatenated in the order in which they were described in the request. For variable lists this is always the order of the list definition. For each variable or a variable in a variable list successfully accessed, the returned data must be the value or the value with object revision of the variable object (if requested). Otherwise it must be an error status. The returned data must also be an error status if the object revision is requested and the variable object accessed does not support the object revision attribute.

Error Status

This parameter indicates the most probable reason why the operation failed using the error class and error code defined for the read service.

Value

This parameter contains a value read. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Value with Object Revision

This parameter contains a value read plus the object revision of the variable object. For variables, this parameter contains the value of the variable. For variable lists, this parameter contains the values of each of the variables in the list concatenated in the order in which they appear in the list.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.4.3.8.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Exchange Service is a "best effort" service. Best effort means that the service succeeds if either of the requested read or write operations succeeds.

The write operation succeeds if the access to any variables of the list or any variable in the variable list succeeds.

The read operation succeeds if any variable accessed in the list or variables in the variable list succeeds.

7.2.5 Event ASE**7.2.5.1 Overview**

The FAL event model defines three event related objects, event objects for the definition of events and their messages, event notifiers for the distribution of event messages, and event lists for the acquisition of event summary information from a group of event objects.

Event objects are used to define messages used to report event occurrences. Event messages contain information that identifies and describes occurrences of events. To simplify the processing of event messages, event objects are defined to generate messages using one of four defined formats.

NOTE For example, event objects may be defined that generate event messages containing a time-tag and/or a message count for event occurrences.

Notifiers are responsible for collecting event messages from event objects, and distributing one or more in a single invocation of the FAL event notification service. The number of event messages that may be submitted in a single service invocation is limited by the maximum APDU size that can be transferred by the AR. Figure 19 illustrates this concept.

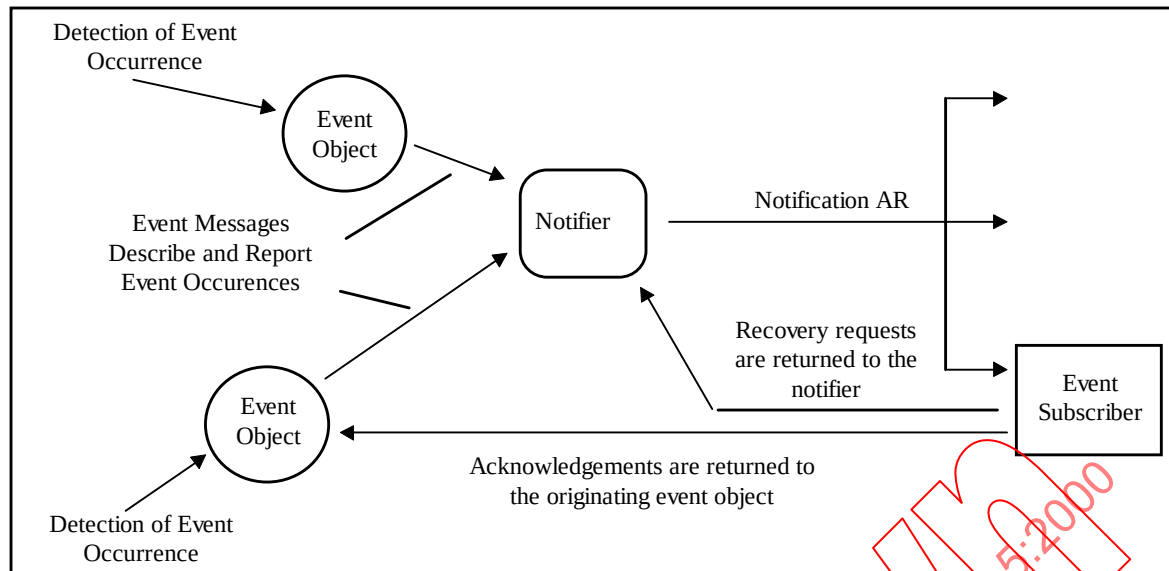


Figure 19 – Event Model Overview

To provide flexibility in the event notification process notifiers may be defined to use one of four types of event notifications to carry event messages. Once defined, a notifier always uses the same type of event notification to carry one or more different types of event messages.

If an application process fails to receive one or more event notifications, a notification recovery service is provided for it to request a retransmission from the notifier.

In addition, the get event summary service is provided to permit an application process to query one or more event objects to select those that meet certain criteria pertaining to the current state of the event objects. An event list object is defined that allows event objects to be grouped together to simplify the query process.

The AR endpoint class used to distribute event notifications is specified by one of the notifier attributes. An event attribute is used to specify which AR distributes the notifications.

Queued ARs are most often used because they prevent event notifications from being overwritten before they are read. When the purpose of the notification is to signal a change in the state of a field device AP, buffered ARs may be used.

NOTE When no events have occurred during a specified period of time notifiers may choose to generate a special "heartbeat" event notification that indicates to subscribers that the notifier is still alive. These special notifications look like the last normal notification sent except that they do not contain the event messages. Heartbeat notifications are defined to permit event subscribers to determine if they have missed an event notification.

In this model, application processes are responsible for providing the functions for event, notifier, and event list objects, and the FAL is responsible for providing communication services designed specifically for them. The application process detects events, builds event messages and aggregates them together. It distributes the aggregated set using the FAL event notification service. At the receiving end, subscriber application processes can use one of two event acknowledgment services to acknowledge event occurrences.

The remainder of this clause specifies the class definitions and services for event, notifier, and event list objects.

7.2.5.2 Event Model Class Specifications

This subclause provides formal class definitions for FAL event objects. Each class definition specifies a type of event class as a combination of its attributes and the FAL services defined for them.

7.2.5.2.1 Event Formal Model

FAL ASE:	EVENT ASE

CLASS: **EVENT**

CLASS ID: 4

PARENT CLASS: TOP

ATTRIBUTES:

1	(m)	Attribute:	Event Status

1.1 (o) Attribute: Active (TRUE,FALSE)

1.2 (o) Attribute: Detected (TRUE,FALSE)

1.3 (m) Attribute: Enabled (TRUE,FALSE)

1.4 (o) Attribute: Acknowledged (TRUE,FALSE)

2	(m)	Attribute:	Message Type	(SIMPLE, REPORTED EVENT COUNT, EVENT DETECTION TIME, COMPOSITE, SIMPLE WITH DATA, REPORTED EVENT COUNT WITH DATA, EVENT DETECTION TIME WITH DATA, COMPOSITE WITH DATA)

3 (m) Attribute: Access Privilege

3.1 (m) Attribute: Password

3.2 (m) Attribute: Access Groups

3.3 (m) Attribute: ~~Access Rights~~

4 (o) Attribute: Last Reported Event Info

4.1 (o) Attribute: Count

4.2 (o) ~~Attribute:~~ ~~Detection Time~~

5 (c) Constraint: Message Type = SIMPLE WITH DATA |
REPORTED EVENT COUNT WITH DATA |
EVENT DETECTION TIME WITH DATA |
COMPOSITE WITH DATA)

5.1 (c) ~~Attribute:~~ Event Data Specifier

5.1.1 (s) Attribute: Variable ID

5.1.2 (s) Attribute: Data Type ID and Length

6 (c) Constraint: Event Status.Acknowledgment Supported = TRUE

6.1 (o) Attribute: Acknowledgment Data Specifier

6.1.1 (s) Attribute: Variable ID

6.1.2 (s) Attribute: Data Type ID and Length

SERVICES:

1	(o)	OpsService:	Acknowledge Event
2	(o)	OpsService:	Acknowledge Event List
3	(o)	OpsService:	Enable Event
4	(o)	OpsService:	Enable Event List
5	(o)	OpsService:	Get Event Summary
6	(o)	OpsService:	Get Event Summary List
7	(o)	OpsService:	Query Event Summary List

7.2.5.2.1.1 Attributes

Event Status

This attribute indicates the status of the event. The status is indicated as a combination of Boolean attributes: active, detected, enabled, acknowledgment support, and acknowledged.

Active

This optional attribute indicates, when TRUE, that events are capable of being detected. If FALSE, event occurrences are not detected.

NOTE The notifier object also has an enable attribute used to control whether or not the event notifications are to be generated.

Detected

This optional attribute indicates, when TRUE, that the conditions for detecting an event occurrence are TRUE.

Enabled

This attribute indicates, when TRUE, that event messages are to be reported to the related notifier upon detection of event occurrences. If FALSE, event messages are not to be generated and reported when event occurrences are detected.

NOTE The notifier object also has an enable attribute used to control whether or not the event notifications are to be generated.

Acknowledged

This optional attribute indicates, when TRUE, that all event messages generated for this event object have been acknowledged. This attribute will also be TRUE if no event messages have been generated or if the event object does not support acknowledgment. The value FALSE indicates that the event object is waiting for an acknowledgment.

Message Type

This attribute specifies which type of event message is generated when an event occurs. The defined messages types and their structure are:

Type	Contents
SIMPLE MSG	Event Numeric ID
REPORTED EVENT COUNT	Event Numeric ID Last Detected Event Info.Count
EVENT DETECTION TIME	Event Numeric ID Last Reported Event Info.Detection Time
COMPOSITE	Event Numeric ID Last Reported Event Info.Count Last Reported Event Info.Detection Time
SIMPLE MSG WITH DATA	Event Numeric ID Event Data Identified by the Event Data Specifier
REPORTED EVENT COUNT WITH DATA	Event Numeric ID Last Detected Event Info.Count Event Data Identified by the Event Data Specifier
EVENT DETECTION TIME WITH DATA	Event Numeric ID Last Reported Event Info.Detection Time Event Data Identified by the Event Data Specifier
COMPOSITE WITH DATA	Event Numeric ID Last Reported Event Info.Count Last Reported Event Info.Detection Time Event Data Identified by the Event Data Specifier

Access Privilege

This attribute specifies the access controls defined for this event. It is composed of the following:

Password

This attribute contains the password for the access rights. Its value is null if it is not used.

Access Groups

This attribute identifies which of the eight user-defined access groups are defined for the event. More than one access group may be defined.

Access Rights

This attribute defines the type of access defined for the event. Valid values are:

- Right to Acknowledge for registered Password
- Right to Enable/Disable for registered Password
- Right to Acknowledge for the Access Groups
- Right to Enable/Disable for the Access Groups
- Right to Acknowledge for all Communication Partners
- Right to Enable/Disable for all Communication Partners

Last Reported Event Info

This optional attribute contains the count and time of detection of the last reported event.

Count

This optional attribute counts the number of reported event occurrences. It is incremented by one for each event occurrence detected and reported. When the event object is disabled, detected event occurrences are not counted.

When the value of this attribute rollover, it rollover to 1, skipping 0. The value 0 is always used to indicate that no event messages have been generated. If the message type attribute is

set to REPORTED EVENT COUNT or COMPOSITE, its updated value is included in event messages.

NOTE The presence of this attribute is independent of whether it is contained in an event message. Its presence indicates this attribute is accessible using the FAL management services. FAL is not involved with incrementing the value of this attribute. Therefore, the FAL does not know when its value rollover, or if it rollover properly.

Detection Time

This optional attribute specifies the time that the last reported event was detected. If the message type attribute is set to EVENT DETECTION TIME or COMPOSITE its value is included in event messages.

NOTE The presence of this attribute is independent of whether it is contained in an event message. Its presence indicates this attribute is accessible using the FAL management services.

Event Data Specifier

This conditional-type attribute specifies user data to be included in an event message. Event data is specified by either identifying a variable object to be incorporated into an event message, or by simply specifying its data type and length. This attribute is present when the Message Type indicates that event data is to be included in event messages.

Variable ID

This selection type attribute identifies a variable object or a variable list object contained in the same AP as the event whose value is to be incorporated into an event message.

Data Type ID and Length

This selection type attribute identifies the data type and length of the user data that is to be incorporated into an event message. Depending on the data type, the length may be defined by the data type.

Acknowledgment Data Specifier

This optional attribute specifies user data to be included in an acknowledgment message. Acknowledgment data is specified by either identifying a variable object to be incorporated into an acknowledgment message, or by simply specifying its data type.

Variable ID

This selection type attribute identifies a variable object or a variable list object contained in the same AP as the event whose value is to be incorporated into an acknowledgment message.

Data Type ID and Length

This selection type attribute identifies the data type and length of the user data that is to be incorporated into an acknowledgment message.

7.2.5.2.1.2 Services

All services defined for this class are optional. When an instance of the class is defined, at least one has to be selected.

Acknowledge Event

This optional confirmed service is provided to permit a subscriber of an event to acknowledge receipt of an event message.

Acknowledge Event List

This optional confirmed service is provided to permit a subscriber of an event to acknowledge receipt of a list of event messages generated by one or more specified event objects.

Enable Event

This optional service is provided to permit a subscriber of an event to enable an event.

Enable Event List

This optional service is provided to permit a subscriber of an event to enable a list of events.

Get Event Summary

This optional service provides the ability to retrieve summary information for one event.

Get Event Summary List

This optional service provides the ability to retrieve summary information for a list of events.

Query Event Summary List

This optional service provides the ability to query a set of events for summary information based on selection criteria.

7.2.5.2.2 Event List Formal Model

FAL ASE:	EVENT ASE
CLASS:	EVENT LIST
CLASS ID:	17
PARENT CLASS:	TOP
ATTRIBUTES:	
1 (m) Attribute:	Number of Entries
2 (m) Attribute:	List Of Events
SERVICES:	
1 (o) OpsService:	Get Event Summary List
2 (o) OpsService:	Query Event Summary List

7.2.5.2.2.1 Attributes**Number of Entries**

This attribute contains the number of events in the list.

List of Events

This attribute identifies the event objects are contained in the event list.

7.2.5.2.2.2 Services

All services defined for this class are optional. When an instance of the class is defined, at least one has to be selected.

Get Event Summary List

This optional service provides the ability to retrieve summary information for a list of events.

Query Event Summary

This optional service provides the ability to query a set of events for summary information based on selection criteria.

7.2.5.2.3 Notifier Formal Model

This subclause provides formal class definitions for notifier objects. Notifiers provide the ability to send event notifications composed of event occurrence messages from one or more event objects.

FAL ASE:		EVENT ASE	
CLASS:		NOTIFIER	
CLASS ID:		18	
PARENT CLASS:		TOP	
ATTRIBUTES:			
1	(m)	Attribute:	Enabled (TRUE,FALSE)
2	(m)	Attribute:	Notification AREP Class
3	(m)	Attribute:	Notification AREP
4	(m)	Attribute:	Notification Type (BASIC, SEQUENCED, TIME OF NOTIFICATION, COMPOUND)
5	(o)	Attribute:	Contained Message Type (SIMPLE, REPORTED EVENT COUNT, EVENT DETECTION TIME, COMPOSITE)
6	(o)	Attribute:	Contained Event Data (TRUE, FALSE)
7	(o)	Attribute:	Last Notification Sequence Number
8	(o)	Attribute:	List Of Events
SERVICES:			
1	(m)	OpsService:	Event Notification
2	(o)	OpsService:	Notification Recovery

7.2.5.2.3.1 Attribute

Enabled

When TRUE, this attribute indicates that the notifier object is enabled. When enabled the notifier object groups event messages from one or more event objects together and submits them in a single invocation of the event notification service to the FAL for conveyance.

When FALSE, the notifier object is disabled. When disabled, the notifier is inactive. It does not issue any event notification service requests.

Notification AREP Class

This attribute identifies the class of the AREP required to convey event notifications.

Notification AREP

This attribute identifies the AREP configured to convey event notifications. This AREP is also the AREP used for reporting the event notifications generated as the result of an event recovery request.

Notification Type

This attribute specifies the type of notification issued by the notifier. The defined notification types are:

Type	Contents
BASIC	Notifier ID
SEQUENCED	Notifier ID Sequence Number
TIME OF NOTIFICATION	Notifier ID Notification Time-Tag
COMPOUND	Notifier ID Sequence Number Time-Tag

Contained Message Type

This attribute specifies the types of event messages that may be contained in notifications issued by the notifier. The event message types are defined by the event object. When an event notification is constructed by this notifier object, the types of event messages that may be contained in the event notification are defined by this attribute. The defined types are:

Contained Message Type	Description
SIMPLE	All event messages in all notifications are of type SIMPLE.
REPORTED EVENT COUNT	All event messages in all notifications are of type REPORTED EVENT COUNT.
EVENT DETECTION TIME	All event messages in all notifications are of type EVENT DETECTION TIME.
COMPOSITE	All event messages in all notifications are of type COMPOSITE.
SIMPLE WITH DATA	All event messages in all notifications are of type SIMPLE.
REPORTED EVENT COUNT WITH DATA	All event messages in all notifications are of type REPORTED EVENT COUNT WITH DATA.
EVENT DETECTION TIME WITH DATA	All event messages in all notifications are of type EVENT DETECTION TIME WITH DATA.
COMPOSITE WITH DATA	All event messages in all notifications are of type COMPOSITE WITH DATA.

Contained Event Data

This attribute, when TRUE, indicates that event messages conveyed by the notifier may contain event data. Event messages for each of the event message types may contain data. This attribute indicates whether or not event messages that are defined for an event object to contain data are capable of being transferred in an event notification generated by this notifier object.

Last Notification Sequence Number

The conditional attribute contains the last sequence number used. It is incremented for each event notification service invocation. When the value of this attribute rolls-over, it rolls-over to 1,

skipping 0. The value 0 is always used to indicate that no event notifications have been generated. It is mandatory when the value of notification type is SEQUENCE or COMPOUND.

NOTE The FAL does not check to see if the roll-over has been performed properly.

List of Events

This optional attribute identifies the events that are configured for this notifier.

7.2.5.2.3.2 Services

Event Notification

This service is provided to permit notifier object to report the occurrence of one or more events in a single service invocation.

Notification Recovery

This optional service is provided to permit one or more event notification subscribers to request that the notifier resend one or more event notifications that it has retained. If this service is supported, then the Sequence Number attribute is required.

7.2.5.3 Event ASE Service Specifications

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

- Acknowledge Event
- Acknowledge Event List
- Enable Event
- Enable Event List
- Event Notification
- Notification Recovery
- Get Event Summary
- Get Event Summary List
- Query Event Summary

7.2.5.3.1 Acknowledge Event

This service allows the acknowledgment of a reported event.

7.2.5.3.1.1 Service Primitives

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

Table 41 –Acknowledge Event

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Key Attribute	M	M(=)		
Reported Event Count	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Reported Event Count

This parameter is used to select a previously generated event message by the reported event count.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.5.3.1.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The association between the event identified in this service and the corresponding event object in the receiving AP is performed by the FAL user.

7.2.5.3.2 Acknowledge Event List Service

This confirmed service is used by an AP receiving an event message to acknowledge receipt of one or more event messages. The service contains an acknowledgment policy parameter that indicates for each specified event message, whether only that event message is to be acknowledged or whether all event messages that were generated by the same event object up to and including the specified event are to be acknowledged. The Acknowledge Event List is a best effort service.

Withdrawing
IECNORM.COM: Click to view the full PDF of IEC 61158-5:2000

7.2.5.3.2.1 Service Primitives

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

Table 42 –Acknowledge Event List Service Parameters

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Acknowledgment Policy	C	C(=)		
Messages to Ack	M	M(=)		
List Of Event ID	M	M(=)		
List Of Message Specifier	U	U(=)		
Reported Event Count	U	U(=)		
Event Detection Time	U	U(=)		
List Of Acknowledgment Data	U	U(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
List Of Access Result			M	M(=)
List Of Error			C	C(=)
Result(-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Acknowledgment Policy

This conditional parameter indicates whether only the specified message is to be acknowledged or whether all messages up to and including the specified message are to be acknowledged. It is present only when the List of Messages to Ack parameter identifies a single message.

Messages to Ack

This parameter contains the messages of all events to be acknowledged.

List of Event ID

This parameter identifies the events being acknowledged by this service.

List of Message Specifier

This optional parameter specifies which messages are to be acknowledged for each event by this service invocation. If this parameter is not present, all messages for the event are to be acknowledged.

If the acknowledgment policy indicates ONLY THE SPECIFIED then only the message identified by this parameter is to be acknowledged.

If the acknowledgment policy indicates UP TO AND INCLUDING, then the message identified by this parameter, and all messages generated before it, are to be acknowledged.

Reported Event Count

This optional parameter is used to select a previously generated event message by the reported event count.

Event Detection Time

This optional parameter is used to select a previously generated event message by the time of the event detection.

List of Acknowledgment Data

This optional parameter contains the user data to be included in an event acknowledgment in addition to that used to identify the event occurrence. The type of this data is specified by the Acknowledgement Data Specifier attribute of the Event Object.

Result(+)

This selection informs the client that the server was able to successfully acknowledge at least one event object among those appearing in the request.

List of Access Result

This parameter contains the failure/success result of all the event objects acknowledged.

List of Error

This parameter indicates the reason for failure for each failed acknowledgement.

Result(-)

This selection indicates that the service request failed.

7.2.5.3.2.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The association between the events referenced by the Event Acknowledgement service and the corresponding event object in the receiving AP is performed by the FAL user.

7.2.5.3.3 Enable Event Service

This service allows the enabling or disabling of the Event Object.

7.2.5.3.3.1 Service Primitives

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

Table 43 – Enable event

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Key Attribute	M	M(=)		
Enable Flag	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the service specific parameters of the service request.

Key Attribute

This parameter identifies the variable or variable list by one of its key attributes.

Enable Flag

This Boolean parameter, when TRUE, indicates that the event object is to be enabled, and when FALSE, that the event object is to be disabled.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.5.3.3.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.5.3.4 Event Notification Service

This unconfirmed service is used by a notifier of an FAL AP to notify other APs that one or more events have occurred. When this service is used without including a list of event messages, the event notification is regarded as a “heartbeat” event notification. This service is unconfirmed.

7.2.5.3.4.1 Service Primitives

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

Table 44 – Event Notification Service Parameters

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Destination DL-Address	C	
Source DL-Address		C
Notifier ID	C	C(=)
Sequence Number	U	U(=)
Notification Time	U	U(=)
List of Event Messages	U	U(=)
List of Event Key Attributes	M	M(=)
List of Event Data Types	C	C(=)
List of EventMessage Contents	C	C(=)
Reported Event Count	C	C(=)
Event Detection Time	C	C(=)
Event Data	C	C(=)

Argument

The argument contains the parameters of the service request.

Destination DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the destination address to which the requested Event Notification is to be sent.

Source DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the source address from which the indicated Event Notification is to be sent.

Notifier ID

This conditional parameter identifies the notifier issuing the event notification. It is present if the AP has more than one notifier defined for it.

Sequence Number

This optional parameter is the sequence number for the event notification. It may be used for notification recovery purposes.

Notification Time

This optional parameter is the time tag for the event notification.

List of Event Messages

This optional parameter contains the list of event messages that are to be reported. It may contain messages from one or more event objects, each containing the same set of parameters (specified by the Contained Message Type attribute of the Notifier object). The contents of each message are specified by its event object and must be consistent with that specified for the Notifier object. When this parameter is not present, the event notification is treated as a “heartbeat” notification.

List of Event Key Attributes

This parameter identifies each of the specific events being acknowledged by this service.

List of Event Data Types

This conditional, optional parameter indicates the data type of each of the event data parameters. This parameter may be present only if the event data parameter is present. If the event data parameter is present, this parameter may be present, but is not required to be.

List of Event Message Contents

This conditional, optional parameter contains the event messages generated by the event objects.

Reported Event Count

This conditional parameter reports the event count for the occurrence being reported. This parameter is present only if it is defined for the message type of the specified event object and if that message type is supported by the specified notifier.

Event Detection Time

This optional parameter reports the time of the event detection. This parameter is present only if it is defined for the message type of the specified event object and if that message type is supported by the specified notifier.

Event Data

This conditional parameter contains user data to be included in an event message in addition to that used to identify the event occurrence. This parameter is present only if it is defined for the specified event object and if it is supported by the specified notifier.

7.2.5.3.4.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

7.2.5.3.5 Enable Event List Service

This service allows the enabling or disabling of the Event Object.

7.2.5.3.5.1 Service Primitives

The service parameters for this service are shown in Table 45.

Table 45 – Enable Event List

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
List Of Key Attributes	M	M(=)		
List Of Enable Flags	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
List Of Enable Result			M	M(=)
List Of Error			C	C(=)
Result(-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the service specific parameters of the service request.

List of Key Attributes

This parameter identifies each event by one of its key attributes.

List of Enable Flags

This parameter, indicates for each event, whether the corresponding event object is to be enabled (when the respective event flag is TRUE), or to be disabled (when the respective event flag is FALSE).

Result(+)

This selection type parameter indicates that the service request succeeded.

List of Enable Result

This list of Boolean parameter indicates for each event that was to be enabled or disabled, if the Enable or Disable action was successful (Enable Result = TRUE) or failed (Enable Result = FALSE).

List of Error

This list of error parameter provides information on the kind of error occurred for each event whose Enable or Disable Action failed.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.5.3.5.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service. The Enable Eventlist is a “best effort” service.

7.2.5.3.6 Notification Recovery Service

This unconfirmed service is used to request that a specified number of retained event notifications be returned. Notifications are returned using the event notification service.

7.2.5.3.6.1 Service Primitives

The service parameters for this service are shown in Table 46.

Table 46 – Notification recovery service parameters

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Destination DL-Address	C	
Source DL-Address		C
Notifier ID	M	M(=)
Number To Recover	U	U(=)
Sequence Number	U	U(=)

Argument

The argument contains the parameters of the service request.

Destination DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the destination address to which the requested notification recovery request is to be sent.

Source DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the source address from which the indicated notification recovery request is to be sent.

Notifier ID

This parameter identifies the notifier to which this service is directed.

Number to recover

This optional parameter specifies the number of event notifications to be re-sent. If this number is greater than the actual number stored, then only the available notifications will be sent. The default for this is one.

Sequence Number

This optional parameter specifies the sequence number of the oldest event notification to be re-sent. If not present, the last notification sent is being requested. It may be used only when the specified notifier uses sequence numbers.

7.2.5.3.6.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

Recovery is performed by the responding AP by re-issuing Event Notification service requests containing the notification messages to be recovered.

7.2.5.3.7 Get Event Summary Service

This confirmed service is used to request summary information for a specified event object.

7.2.5.3.7.1 Service Primitives

The service parameters for this service are shown in Table 47.

Table 47 – Get Event Summary Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Event ID	M	M(=)		
Recovered Message Requested	U	U(=)		
Object Revision Requested	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Event Status			M	M(=)
Active			M	M(=)
Detected			M	M(=)
Enabled			M	M(=)
Acknowledgment Supported			M	M(=)
Acknowledged			M	M(=)
Object Revision			C	C(=)
Recovered Message			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Event ID

This mandatory parameter identifies the event object for which the event summary is being requested. An event list object cannot be specified because the members of the list may be dynamically changed by the AP, making it necessary to identify each event in the response.

Recovered Message Requested

This optional parameter specifies, when TRUE, that the last message generated by the event is to be returned.

Object Revision Requested

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

Result(+)

This selection indicates that the service request succeeded.

Event Status

This parameter contains the values of the event status attribute. It indicates the status of the event. The status is indicated as a combination of Boolean parameters: active, detected, enabled, acknowledgment support, and acknowledged.

Detected

This Boolean parameter indicates, when TRUE, that the conditions for detecting an event occurrence are TRUE.

Active

This Boolean parameter indicates, when TRUE, that events are capable of being detected. If FALSE, event occurrences are not detected.

Enabled

This Boolean parameter indicates, when TRUE, that event messages are to be reported to the related notifier upon detection of event occurrences. If FALSE, event messages are not to be generated and reported when event occurrences are detected.

Acknowledgment Supported

This Boolean parameter indicates, when TRUE, that this event object supports the acknowledgment of event messages.

Acknowledged

This Boolean parameter indicates, when TRUE, that all event messages generated for this event object have been acknowledged. This attribute will also be TRUE if no event messages have been generated or if the event object does not support acknowledgment. The value FALSE indicates that the event object is waiting for an acknowledgment.

Recovered Message

This conditional parameter is present if the service request specified message recovery was requested. When present, this parameter contains the recovered event message.

Object Revision

This conditional parameter contains the object revision of the specified object. It is present if the object revision was requested.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.5.3.7.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Get Event Summary Service is an "all or nothing" service that operates through a queue or buffer. All or nothing means that the service succeeds only if the requested summary is returned.

IECNORM.COM: Click to view the full PDF of IEC 61158-5:2000

Withdrawn

7.2.5.3.8 Get Event Summary List Service

7.2.5.3.8.1 Service Primitives

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

Table 48 – Get Event Summary List Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Summary Requests	M	M(=)		
Requested Events	M	M(=)		
All Events	S	S(=)		
List of Events	S	S(=)		
Recovered Message Requested	U	U(=)		
Object Revision Requested	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Event Summaries			M	M(=)
List Of Event ID			C	C(=)
List Of Access Result			M	M(=)
List Of Response			M	M(=)
Error Status			S	S(=)
Event Summary			S	S(=)
Event Status			M	M(=)
Active			M	M(=)
Detected			M	M(=)
Enabled			M	M(=)
Acknowledgment Supported			M	M(=)
Acknowledged			M	M(=)
Object Revision			C	C(=)
Recovered Message			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Summary Requests

This mandatory parameter specifies a list of events, and the summary information for each, that is being requested. Each entry in the list may identify either an event or an event list object. Each entry in the list also identifies the summary information that is being requested. If an entry identifies an event list, then the summary information requested will apply to all events in the event list.

Requested Events

This mandatory parameter indicates that summaries are being requested for all events, or it contains a list that identifies the event objects for which event summaries are being requested.

All Events

This selection type parameter indicates that summaries are being requested for all events.

List of Events

This selection type parameter is a list that contains the identifiers of the events and/or event lists for which event summaries are being requested. When selected, the list must contain at least one entry.

Recovered Message Requested

This optional parameter specifies, when TRUE, that the last message generated by the event is to be returned.

Object Revision Requested

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

Result(+)

This selection indicates that the service request succeeded.

Event Summaries

This parameter contains a response for each requested event. The order of the information is the same as the order in which the objects were referenced in the service request.

List of Event ID

This conditional parameter identifies each event for which an event summary or error status is returned.

List of Access Result

This list of Boolean parameters provides for each event for which an event summary was requested an information, whether the access was successful (Access Result = TRUE) or not (Access Result = FALSE).

List of Response

This parameter contains an error status indicator or the event summary for each requested event. The order of the information is the same as the order in which the objects were referenced in the service request.

Error Status

This selection type parameter specifies if the condition which triggers the reporting of the event is active. A value of TRUE indicates that the underlying condition which triggers the event is active. A value of FALSE indicates that the underlying condition which triggers the event is not active.

Event Summary

This selection type parameter contains the summary information for the specified event.

Event Status

This parameter contains the values of the event status attribute. It indicates the status of the event. The status is indicated as a combination of Boolean parameters: active, detected, enabled, acknowledgment support, and acknowledged.

Detected

This Boolean parameter indicates, when TRUE, that the conditions for detecting an event occurrence are TRUE.

Active

This Boolean parameter indicates, when TRUE, that events are capable of being detected. If FALSE, event occurrences are not detected.

Enabled

This Boolean parameter indicates, when TRUE, that event messages are to be reported to the related notifier upon detection of event occurrences. If FALSE, event messages are not to be generated and reported when events occurrences are detected.

Acknowledgment Supported

This Boolean parameter indicates, when TRUE, that this event object supports the acknowledgment of event messages.

Acknowledged

This Boolean parameter indicates, when TRUE, that all event messages generated for this event object have been acknowledged. This attribute will also be TRUE if no event messages have been generated or if the event object does not support acknowledgment. The value FALSE indicates that the event object is waiting for an acknowledgment.

Recovered Message

This conditional parameter is present if the service request specified message recovery was requested. When present, this parameter contains the recovered event message.

Object Revision

This conditional parameter contains the object revision of the specified object. It is present if the object revision was requested.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.5.3.8.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Get Event Summary List Service is a "best effort" service that operates through a queue or buffer. Best effort means that the service succeeds if at least one event summary is returned. If the user is not able to read at least one of the values, the service fails and the user issues a Get Event Summary List Service response (-) primitive indicating the reason.

7.2.5.3.9 Query Event Summary List Service

This confirmed service provides the means for an application to request that all events, an event list, or a list of events be searched according to the criterion provided. The list of valid criteria supported by the responding application process is defined by the List of Event Summary Selection Criteria attribute of the responding AP. A supported criterion may define either an individual event characteristic, or a combination of event characteristics.

NOTE The selection criteria attribute defined for an event may be based on an event characteristic such as its status, whether it is enabled or disabled, or whether it has been acknowledged. It is also possible to specify user-defined criteria such as the events related to a particular operating unit or a piece of equipment.

Withdrawing
IECNORM.COM: Click to view the full PDF of IEC 61158-5:2000

7.2.5.3.9.1 Service Primitives

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

Table 49 – Query Event Summary List Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Events to Search	M	M(=)		
All Events	S	S(=)		
List of Events	S	S(=)		
Selection Criterion	M	M(=)		
Recovered Message Requested	U	U(=)		
Object Revision Requested	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Event Summaries			M	M(=)
List Of Event ID			C	C(=)
List Of Access Result			M	M(=)
List Of Response			M	M(=)
Event Status			M	M(=)
Active			M	M(=)
Detected			M	M(=)
Enabled			M	M(=)
Acknowledgment Supported			M	M(=)
Acknowledged			M	M(=)
Object Revision			C	C(=)
Recovered Message			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Events to Search

This parameter identifies the event objects that are to be searched. The events are identified by specifying all events, an event list, or a list of events.

All Events

This selection type parameter indicates that all events are to be searched.

List of Events

This selection type parameter specifies events that are to be searched. The events may be specified as a list of event objects, a list of event list objects, or a combination of the two. When selected, the list must contain at least one entry.

Selection Criterion

This parameter identifies the criterion to use in the selection process. The value used must be contained in the List of Event Summary Selection Criteria attribute of the AP containing the events to be searched.

Recovered Message Requested

This optional parameter specifies, when TRUE, that the last message generated by the event is to be returned.

Object Revision Requested

This optional parameter is used to request that the value of the object revision attribute be returned. A value of TRUE indicates that object revision is desired. A value of FALSE indicates that object revision is not desired. If the Object Revision is requested, but is not supported for the specified object, the service fails.

Result(+)

This selection indicates that the service request succeeded.

Event Summaries

This parameter contains the summary information for each selected event

List of Event ID

This conditional attribute is present when the selection criterion is not "none" or when the selection criterion is "none" and the specified events are "all" or a "list of events".

List of Access Result

This list of Boolean parameters provides for each event for which an event summary is returned an information, whether the access was successful (Access Result = TRUE) or not (Access Result = FALSE).

List of Response

This parameter contains an error status indicator or the event summary for each returned event summary. The order of the information is the same as the order in which the objects were referenced in the service request.

Event status

This parameter specifies, when TRUE, that the conditions for detecting an event are currently TRUE. This parameter has the value FALSE if the conditions are FALSE or if the event object is not active.

Active

This parameter specifies, when TRUE, that event detection is active for the event. When active, the event object detects when events occur. When inactive, it does not.

Enabled

This parameter specifies the value of the Enabled attribute of the identified Event object.

Unacknowledged

This parameter, when TRUE, specifies that the event object does not have outstanding event occurrences for which it is awaiting acknowledgments, or that it does not support acknowledgment. If the Event is waiting for acknowledgments this parameter has a value of FALSE.

Recovered Message

This conditional parameter is present if the service request specified message recovery was requested. When present, this parameter contains the recovered event message.

Object Revision

This conditional parameter is present if the service request specified the object revision was requested. When present, this parameter contains the object revision of the event object selected.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.5.3.9.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

The Query Event Summary Service is a "best effort" service that operates through a queue or buffer. Best effort means that the service succeeds if at least one event summary is returned. If the user is not able to obtain the summary for at least one event, the service fails and the user issues a Query Event Summary Service response (-) primitive indicating the reason.

7.2.6 Load Region ASE

7.2.6.1 Overview

A Load Region represents an unstructured memory area whose contents may be uploaded (read) or downloaded (written). Unstructured in this context means that the memory area is represented only as an ordered sequence of octets. No other structure is apparent.

Load regions may represent an unnamed volatile memory area, such as that implemented by dynamic computer memory, or a named non-volatile memory object, such as a file. The contents of a load region are referred to as a load image. Load images may contain programs or data. The transfer of a load image to or from a load region is performed using the load process.

To maintain integrity, only one load process for a load region is permitted at a time. The Load Region State attribute of the load region is used to indicate whether the load region is empty, being downloaded, or loaded (see the definition of the attribute for a complete list of the states). Load regions may be cleared using the Discard service.

The Upload State attribute indicates the progress of an upload. This attribute is defined to separate the state of the contents of the load region from the operation of having the contents uploaded (dumped).

This load region model provides services that permit an AP to initiate the download or upload of one of its load regions. It identifies the load region and whether to upload or download as parameters of the request. A third parameter indicates whether the transfer of load image segments is to be accomplished using the Pull Segment service or the Push Segment service.

NOTE The Push and Pull Segment services have no relationship to the Push and Pull AREP types.

Load image segments are pulled by having the receiver of the load image issue requests for them. The AP that contains the load image responds by returning the requested segment, indicating with a parameter when the final segment of the load image has been returned.

Load image segments are pushed by having the AP containing the load image send individual segments to the receiver and wait for a response that indicates whether the segment was received. In this case, the sender indicates when the last segment has been transferred.

After the last segment has been received, the AP that initiated the transfer ends the load process by issuing a terminate request to the remote AP. The terminate request may also be used by either AP when an AP determines that the load process cannot be completed successfully.

7.2.6.2 Load Region Model Specification

Load Regions are modeled as server objects. The Load Region Model below specifies the attributes and services used by client APs to download and upload load regions. The specification of the client AP is beyond the scope of this standard.

7.2.6.2.1 Load Region Formal Model

FAL ASE: LOAD REGION ASE

CLASS: LOAD REGION

CLASS ID: 2

PARENT CLASS: TOP

ATTRIBUTES:

1	(m)	Attribute:	Load Region Size
2	(m)	Attribute:	Access Privilege
2.1	(m)	Attribute:	Password
2.2	(m)	Attribute:	Access Groups
2.3	(m)	Attribute:	Access Rights
3	(m)	Attribute:	Local Address
4	(m)	Attribute:	Load Region State
5	(m)	Attribute:	Upload State
6	(m)	Attribute:	Number of Related Objects In Use
7	(o)	Attribute:	List of Related Objects
7.1	(m)	Attribute:	ClassID
7.2	(m)	Attribute:	Numeric ID
7.2	(m)	Attribute:	In Use Flag (TRUE, FALSE)
8	(o)	Attribute:	Contents Size
9	(o)	Attribute:	Load Image Name
10	(o)	Attribute:	Fixed Length Segments (TRUE/FALSE) Default: FALSE
11	(c)	Constraint:	Fixed Length Segments = TRUE
11.1	(o)	Attribute:	Fixed Segment Length
12	(o)	Attribute:	Intersegment Request Timeout
13	(o)	Attribute:	Sharable
14	(o)	Attribute:	Local Detail

SERVICES:

1	(m)	Ops Service:	Initiate Load
2	(o)	Ops Service:	Push Segment
3	(o)	Ops Service:	Pull Segment
4	(m)	Ops Service:	Terminate Load
5	(o)	Ops Service:	Discard

7.2.6.2.1.1 Attributes

Load Region Size

This attribute specifies the maximum size of the Load Region in octets.

Access Privilege

This attribute specifies the access controls defined for this load region. It is composed of the following:

Password

This attribute contains the password for the access rights. Its value is null if it is not used.

Access Groups

This attribute identifies which of the eight user-defined access groups are defined for the load region. More than one access group may be defined.

Access Rights

This attribute defines the type of access defined for the load region. Valid values are:

- Right to Use in a FI for the Access Groups
- Right to Write for the Access Groups
- Right to Read for the Access Groups
- Right to Use in a FI for the registered Password
- Right to Write for the registered Password
- Right to Read for the registered Password
- Right to Use in a FI for all Communication Partners
- Right to Write for all Communication Partners
- Right to Read for all Communication Partners

Local Address

This attribute is a locally significant address of the load region. The value FFFFFFFF hex indicates that no local address is available.

Load Region State

This attribute specifies the state of the Load Region Object. The values of this attribute are:

NOT DOWNLOADABLE	This state indicates that the Load Region is not capable of being downloaded.
DOWNLOADABLE	This state indicates that the Load Region is empty, but is capable of being downloaded.
DOWNLOADING	This state indicates that the download sequence has been initiated.
DOWNLOAD FAILURE	This transient state indicates that the download sequence has failed, but has not yet been terminated.
DOWNLOAD SUCCESS	This transient state indicates that the download sequence has succeeded, but has not yet been terminated.
LOADED	This state indicates that the download sequence has terminated successfully.
IN-USE	This state indicates that the Load Region is loaded and is currently being used.

Download State attribute constraints:

A Load Region object whose contents are stored in non-erasable memory cannot support downloading transfers. Its valid state values are LOADED and IN USE

A Load Region object whose contents are stored in erasable memory supports downloading transfers. Its valid state values are DOWNLOADABLE, DOWNLOADED, DOWNLOADING, DOWNLOADING_FAILURE, DOWNLOADING_SUCCESS, LOADED and IN_USE,

NOTE The state NOT DOWNLOADABLE state is reserved for Load Regions whose contents are stored in erasable memory, but who have encountered a temporary condition that does not allow them to be downloaded. contents of Load Region objects located in an Application Process paying client role.

Upload State

This attribute specifies the state of the Upload Process of the Load Region.

NOT UPLOADABLE	This state indicates that the Load Region is not capable of being uploaded.
UPLOADABLE	This state indicates that the Load Region is capable of being uploaded.
UPLOADING	This state indicates that the upload sequence has been initiated.
COMPLETING UPLOAD	This transient state indicates that the upload sequence has completed, but has not yet been terminated.

Upload State attribute constraints:

A Load Region object whose contents are stored in erasable memory or not support Uploading transfers. Its valid state values are UPLOADABLE, UPLOADING, and COMPLETING UPLOADING

NOTE The NOT UPLOADABLE state is reserved Load Regions whose contents are stored in erasable memory, but who have encountered a temporary condition that does not allow them to be uploaded.

Number of Related Objects in Use

This attribute indicates the number of related Action objects or Function Invocation objects which currently use the contents of the Load Region object.

This attribute is incremented each time a related Function Invocation object enters the RUNNING state or a related Action object is invoked.

This attribute is decremented each time a related Function Invocation object leaves the RUNNING state or a related Action object completes execution.

If the attribute sharable = TRUE, then several Function Invocation objects and Action objects may share the Load Region contents; otherwise only one Function Invocation or Action object can be attached and use the contents of the object

List of Related Objects

This optional attribute specifies the Function Invocation and Action objects sharing the contents of the Load Region. Each Function Invocation object or Action object is identified by specifying its class identifier and its numeric identifier.

This list is respectively increased/decreased each time a related Function Invocation object or Action object is created or deleted.

Class ID

This attribute is the class identifier for the related object.

Numeric ID

This attribute is the numeric identifier for the related object.

In Use Flag

This attribute is TRUE if the related object is a Function Invocation object in the RUNNING state or if the related object is an Action object that is currently executing.

Contents Size

This optional attribute specifies the length of the data contained in the Load Region. The length is indicated in octets. If the state of the load region is DOWNLOADABLE, then the value of this attribute is zero.

Load Image Name

This optional attribute specifies the name of load image contained in the load region.

Fixed-Length Segments

This optional attribute specifies, when TRUE, that the load region supports the upload and download of fixed length segments only. The default value for this attribute is FALSE.

Fixed-Segment Length

This conditional attribute specifies the length in octets of the fixed-length segments uploaded to and downloaded from this load region. The attribute is present when the Fixed Length Segments attribute is TRUE.

Intersegment Request Timeout

This optional attribute specifies the timeout period for receiving segment requests, in seconds.

Sharable

This attribute indicates, when TRUE, that the Load Region object contents may be used by multiple Function Invocation/Action objects. The default value for this attribute is TRUE.

Local Detail

This attribute contains additional information, such as a list of capabilities and/or the number of octets to be transferred.

7.2.6.2.1.2 Services

Initiate Load

This service is used by an application to initiate the download or upload of a Load Region. A parameter of the service indicates whether the load image will be transferred into the Load Region using the Pull Segment or from the Load Region using the Push Segment service.

Push Segment

This optional service is used to transfer load segments in the request/indication primitives. Although this service is optional, at least one of the segment transfer services, Push Segment or Pull Segment, is required.

Pull Segment

This optional service is used to request that a load segment be returned in the response/confirmation primitives. Although this service is optional, at least one of the segment transfer services, Push Segment or Pull Segment, is required.

Terminate Load

This service is used by an AP to terminate the load process.

Discard

This optional service is used to clear the contents of the load region.

7.2.6.3 Load Region Service Specification

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

- Initiate Load
- Terminate Load
- Push Segment
- Pull Segment
- Discard

7.2.6.3.1 Initiate Load Service

This confirmed service is used to request the download or upload of a load region. A parameter of the service indicates whether the load image is to be transferred by the requester using the Pull Segment or Push Segment service. When a download is being requested for a load region in a remote AP (identified by the called load region key attribute) that does not yet exist, the responding AP may either dynamically create the load region and return a positive response, or return a negative response that indicates that the load region does not exist.

7.2.6.3.1.1 Service Primitives

The service parameters for this service are shown in Table 50.

Table 50 – Initiate Load Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Load Region Key Attribute	M	M(=)		
Calling	S	S(=)		
Called	S	S(=)		
Load Type	M	M(=)		
Load Service To Use	M	M(=)		
Load Service Initiator	M	M(=)		
Additional Information	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Additional Detail			U	U(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)
Called Size			C	C(=)

Argument

The argument contains the parameters of the service request.

Load Region Key Attribute

This parameter identifies the load region to be downloaded or to be uploaded.

Calling

This selection type parameter identifies the load region of the requester. It is selected if the requester is the server.

Called

This selection type parameter identifies the load region of the responder. It is selected if the requester is the client.

Load Type

This parameter specifies whether the load is an UPLOAD (out of the load region) or a DOWNLOAD (into the load region).

Load Service to Use

This parameter specifies that the load region is to be transferred from the remote AP using either the Pull Segment or the Push Segment service.

Load Service Initiator

This parameter specifies, when TRUE, that the load service identified by the previous parameter is to be initiated by the AP initiating this service. When FALSE, the remote AP is to initiate the load service.

Additional Information

This optional parameter contains additional information, such as a file name and/or the number of octets to be transferred. It may be present if the requester is the server.

Result(+)

This selection type parameter indicates that the service request succeeded.

Additional Detail

This optional parameter contains additional information, such as a list of capabilities and/or the number of octets to be transferred. It may be present if the responder is the server.

Result(-)

This selection type parameter indicates that the service request failed.

Called Size

This conditional parameter specifies the maximum size in octets that the responding AP is prepared to transfer. It is present if the error was due to a size problem.

7.2.6.3.1.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.6.3.2 Terminate Load Service

This confirmed service is used to terminate the load process. It may be used upon successful completion of the load, or to abort a load in progress. It may be requested by either endpoint.

If the load process has completed, it is always issued by the AP that issued the initiate service that started the load process. If the load process is being aborted,

- a) the load region sending the image returns to the UPLOADABLE Upload State;
- b) the load region receiving the image deletes the incomplete load from its load region and returns to the DOWNLOADABLE State.

7.2.6.3.2.1 Service Primitives

The service parameters for this service are shown in Table 51.

Table 51 – Terminate Load Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Load Region Key Attribute	M	M(=)		
Load Type	M	M(=)		
Load Service Used	M	M(=)		
Terminate Reason	C	C(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Terminate Reason			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Load Region Key Attribute

This parameter specifies one of the key attributes of the Load Region for which the load is being terminated.

Load Type

This parameter indicates the type of load being terminated. Valid values are UPLOAD and DOWNLOAD.

Load Service Used

This parameter indicates the type of load service used to transfer load segments. Valid values are PUSH and PULL.

Terminate Reason

This conditional parameter indicates the success or failure of the load process. It is present if the server issues the terminate request.

Result(+)

This selection type parameter indicates that the service request succeeded.

Terminate Reason

This conditional parameter indicates the success or failure of the load process. It is present if the client issues the terminate request to terminate a download.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.6.3.2.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.6.3.3 Push Segment Service

This confirmed service is used after the load process has been initiated to transfer a single load image segment in its request and indication primitives. The response and confirmation primitives are used to convey the success or failure of the segment transfer. This service may be used to support uploads and downloads.

7.2.6.3.3.1 Service Primitives

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

Table 52 – Push Segment Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Load Type	M	M(=)		
Load Region Key Attribute	M	M(=)		
Segment Number	U	U(=)		
Load Data	M	M(=)		
More Follows	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Load Type

This parameter specifies whether the load is an UPLOAD (out of the load region) or a DOWNLOAD (into the load region).

Load Region Key Attributes

This parameter specifies one of the key attributes of the Load Region whose image is to be transferred.

NOTE The load region is local if this service is being used for an upload. The load region is remote if this service is being used for a download.

Segment Number

This optional parameter identifies the number of the segment which is being transferred. Segment numbers are ascending integers beginning with the value one (1).

Load Data

This parameter contains the data to be loaded.

More Follows

This parameter indicates whether or not any additional data remains to be transmitted.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.6.3.3.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.6.3.4 Pull Segment Service

This confirmed service is used after the load process has been initiated to request that a specified load image segment be returned in the response and confirmation primitives. This service may be used to support uploads and downloads.

7.2.6.3.4.1 Service Primitives

The service parameters for this service are shown in Table 53.

Table 53 – Pull Segment Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Load Type	M	M(=)		
Load Region Key Attribute	M	M(=)		
Segment Number	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Load Data			M	M(=)
More Follows			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Load Type

This parameter specifies whether the load is an UPLOAD (out of the load region) or a DOWNLOAD (into the load region).

Load Region Key Attribute

This parameter specifies one of the key attributes of the Load Region whose image is to be transferred.

NOTE The load region is remote if this service is being used for an upload. The load region is local if this service is being used for a download.

Segment Number

This optional parameter identifies the number of the segment which is being requested. Segment numbers are ascending integers beginning with the value one (1).

Result(+)

This selection type parameter indicates that the service request succeeded.

Load Data

This parameter contains the data to be downloaded or uploaded.

More Follows

This parameter indicates whether or not any additional data remains to be transmitted.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.6.3.4.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.6.3.5 Discard Service

This confirmed service is used to request that the contents of a load region be discarded.

7.2.6.3.5.1 Service Primitives

The service parameters for this service are shown in Table 54.

Table 54 – Discard Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Load Region Key Attribute	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Load Region Key Attribute

This parameter specifies one of the key attributes of the Load Region object whose contents are to be discarded.

Result(+)

This selection indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

7.2.6.3.5.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.6.4 Load Region State Machines

This standard defines state machines for three types of transfers.

Pull_Uploading: The load region contents are retrieved from a Load Region object, segment by segment, by a Client AP.

Pull_Downloading: A load image contained in a Client AP is copied into a Load Region object, segment by segment, at the request of the server AP that contains the load region. That is, the Server AP requests the client to send the segments of the load image. This kind of transfer is allowed for erasable load regions. Its state diagram and transitions table give additional information related to a non-erasable load region contents.

Push_Downloading: A load image contained in a client AP are sent to the Load Region object, segment by segment, by the client AP. Upon receipt of each segment, the Server AP that contains the load region copies it into the load region. This kind of transfer is allowed for erasable load regions. Its state diagram and transitions table give additional information related to a non-erasable load region contents.

Other kinds of transfer are beyond the scope of this standard.

7.2.6.4.1 Pull Upload State Machine

This kind of transfer uses the following services: Initiate Load, Pull Segment and Terminate Load

A Server may initiate an uploading transfer. In this case it issues an Initiate_Load Req service primitive to a Client. This primitive tells the Client to initiate uploading transfer for a specified Load Region. Once the client has pulled the contents, it issues an Initiate_Load.Rsp(+/-) to the Server.

A Client may initiate an uploading transfer. In this case it issues an Initiate_Load Req primitive to a Server. This primitive tells the Server that the contents of a specified load region object will be pulled. Once the client has pulled the contents, it ends the transfer by issuing a Terminate_Load Req to the Server.

7.2.6.4.1.1 Sequencing of Service Primitives

Table 55 summarizes the sequencing of service primitives. In the table below, T1 and T2 show different options for initiating the upload, one by the server AP (T1) and one by the client AP (T2).

Table 55 – Pull Upload Sequencing of Service Primitives

FAL User(Client)		FAL User(Server) Owner of called Load Region
Initiate_Load.Ind	←----(T1)-----	Initiate_Load Req
Initiate_Load Req	----- (T2) ----→	Initiate_Load.Ind (Transfer started)
Initiate_Load.Cnf	←----(T2)-----	Initiate_Load.Rsp
Pull_Segment Req	----- (T3) ----→	Pull_Segment.Ind (Segments Transferred)
Pull_Segment.Cnf	←----(T3)-----	Pull_Segment.Rsp
Terminate_Load Req	----- (T4) ----→	Terminate_Load.Ind (Transfer ended)
Terminate_Load.Cnf	←----(T4)-----	Terminate_Load.Rsp
Initiate_Load.Rsp	----- (T1) ----→	Initiate_Load.Cnf

7.2.6.4.1.2 Service Parameter Value Constraints

Table 56 summarizes the service parameter constraints.

Table 56 – Pull Upload Service Parameter Constraints

Transition Number	Parameter constraints
T1	Initiate_Load Calling Load Region Key attribute selected, Load Type : = upload, Service To Use : = Pull_Segment, Load Service Initiator : = FALSE
T2	Initiate_Load Called Load Region Key Attribute selected, Load Type : = upload Service To Use : = Pull_Segment Load Service Initiator : = TRUE
T3	Pull_Segment Load Type : = upload, Segment Number : = ordered 1 to n if any More Follows : = TRUE/FALSE with FALSE for the last segment
T4	Terminate_Load Load Type : = upload, Load Service Used : = Pull_Segment

7.2.6.4.1.3 Upload State Transition Table

Table 57 gives the upload state transitions.

Table 57 – Pull Upload State Table

Transition	Current State	Event/Action	Next State
1	UPLOADABLE	Initiate_Load.Ind &&Load Region State of this object is in state DOWNLOADABLE II LOADED II IN USE Initiate_Load.Rsp(+) the load region image is segmented in accordance with fixed-length attribute constraints into n segments Segments Left To Transfer : = n -- (local counter)	UPLOADING
2	UPLOADING	Pull_Segment.Ind && Segments Left To Transfer > 1 &&service succeeds(e.g. compliance with Intersegment Req Time Out attribute constraint if supported) Pull_Segment.Rsp(+) Segment Number : = Segments Left To Transfer (if any) with More Follows : = TRUE Segments Left To Transfer : = -1	UPLOADING
3 Exist if supports: Intersegmnt Req Time Out	UPLOADING	Pull_Segment.Ind &&Segments Left To Transfer = any value &&service fails(e.g. no compliance with Intersegment Req Time Out attribute constraint if supported) Pull_Segment.Rsp(-)	UPLOADING
4	UPLOADING	Pull_Segment.Ind &&Segments Left To Transfer=1 (last segment) &&service succeeds(e.g. compliance with Intersegment Req Time Out attribute constraint) Pull_Segment.Rsp(+) Segment Number : = Segments Left To Transfer (if any) More Follows : = FALSE Segments Left To Transfer : = -1	COMPLETING UPLOAD
5	UPLOADING	Terminate_Upload.Ind Terminate_Upload.Rsp(-) Error Info : = object state conflict	UPLOADABLE
6	UPLOADING or COMPLETING UPLOAD	Abort.Ind	UPLOADABLE
7	COMPLETING UPLOAD	Terminate_Upload.Ind Terminate_Upload.Rsp(+)	UPLOADABLE
8	UPLOADABLE	Initiate_Load.Ind && Load Region State attribute is not in state: DOWNLOADABLE II LOADED II IN USE Initiate_Load.Rsp(-) with Error Info : = object constraint conflict	UPLOADABLE

7.2.6.4.2 Pull Download State Machine

This kind of transfer uses the following services: Initiate Load, Pull Segment and Terminate Load.

A Server may initiate a downloading transfer. In this case it issues an Initiate_Load.Req service primitive to a Client. This primitive tells the Client to initiate downloading transfer for a specified Load Region object. Once the Server has pulled the contents, the Client issues an Initiate_Load.Rsp(+/-) to the Server.

A Client may initiate the downloading transfers. In this case it issues an Initiate_Load.req primitive to a Server. This primitive tells the Server to pull the contents of a specified load region object. Once the Server has pulled the contents, it ends the transfer by issuing a Terminate_Load.Reg to the Client

7.2.6.4.2.1 Sequencing of Service Primitives

Table 58 summarizes the sequencing of service primitives.

Table 58 – Pull Download Sequencing of Service Primitives

FAL User(a Client)		FAL User(a Server)
Owner of a calling Object		Owner of called Object
Initiate_Load.Ind	←----(T 1)-----	Initiate_Load.Reg
Initiate_Load.Reg	----- (T2) ---->	Initiate_Load.Ind (Transfer started)
Initiate_Load.Cnf	←----(T2)-----	Initiate_Load.Rsp
Pull_Segment.Ind	<----- (T3) ----	Pull_Segment.Reg (Transfer continued)
Pull_Segment.Rsp	----- (T3) ---->	Pull_Segment.Cnf
Terminate_Load.Ind	<----- (T4) ----	Terminate_Load.Reg (Transfer ended)
Terminate_Load.Rsp	----- (T4) ---->	Terminate_Load.Cnf
Initiate_Load.Rsp	----- (T1) ---->	Initiate_Load.Cnf

7.2.6.4.2.2 Service Parameter Constraints

Table 59 summarizes the service parameter constraints.

Table 59 – Pull Download Service Parameter Constraints

Transition Number	Parameter constraints
T1	Initiate_Load: Calling Load Region Key attribute selected, Load Type : = download, Service To Use : = Pull_Segment, Load Service Initiator : = TRUE
T2	Initiate_Load Called Load Region Key Attribute selected, Load Type : = download Service To Use : = Pull_Segment Load Service Initiator : = FALSE
T3	Pull_Segment Load Type : = download, Segment Number : = ordered 1 to n if any More Follows : = TRUE/FALSE with FALSE for the last segment
T4	Terminate_Load Load Type : = download, Load Service Used : = Pull_Segment Terminate Reason contained in the request

7.2.6.4.2.3 Download State Transition Table

Table 60 gives the download state transitions.

Table 60 – Pull Download State Table

Transition	Current State	Event/Action	Next State
1	DOWNLOADABLE	Initiate_Load.Ind &&Upload State = UPLOADABLE Initiate_Load.Rsp(+) Pull_Segment.Reg	DOWNLOADING
2	DOWNLOADABLE	Initiate_Load.Ind &&Upload State <>UPLOADABLE Initiate_Load.Rsp(-) Error Info : = object constraint conflict	DOWNLOADABLE
3	DOWNLOADING	Pull_Segment.Cnf(+) &&More Follows=TRUE Store received data Pull_Segment.Reg	DOWNLOADING
4	DOWNLOADING	Pull_Segment.Cnf(+) &&More Follows=FALSE Store received data Terminate_Load.Reg Terminate Reason : = success	DOWNLOAD SUCCESS
5	DOWNLOADING	Pull_Segment.Cnf(-) Discard received data Terminate_Load.Reg Terminate Reason : = failure	DOWNLOAD FAILURE
6	DOWNLOADING, DOWNLOAD FAILURE, DOWNLOAD SUCCESS	Abort.Ind Discard received data	DOWNLOADABLE
7	DOWNLOAD FAILURE	Terminate_Load.Cnf(+/-) Discard received data	DOWNLOADABLE
8	DOWNLOAD SUCCESS	Terminate_Load.Cnf(-) Discard received data	DOWNLOADABLE
9	DOWNLOAD SUCCESS	Terminate_Load.Cnf(+) Update Contents Size attribute if present	LOADED
10	LOADED	Initiate_Load.Ind &&Upload state attribute in state UPLOADABLE &&service succeeds(e.g erasable load region contents) Initiate_Load.Rsp(+)	DOWNLOADING
11	LOADED	Discard.Ind &&Upload State attribute in state UPLOADABLE &&service succeeds(e.g. erasable load region contents) Discard.Rsp(+)	DOWNLOADABLE
12	IN USE	Number of Related object In Use attribute incremented &&its new value > 1 Set Related Object In Use flag value to TRUE if present (Start/Resume)	IN USE

(continued on page 194)

Table 60 (concluded)

13	IN USE	Number of Related Object In Use attribute decremented &&its new value >0 Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill)	IN USE
14	LOADED	Initiate_Load.Ind &&Upload State attribute in state <> UPLOADABLE If Service fails(e.g. non erasable load region contents) Initiate_Load.Rsp(-) Error Info : = object constraint conflict	LOADED
15	LOADED	Discard.Ind && Upload State attribute in state <> UPLOADABLE If Service fails(e.g. non erasable load region contents) Discard.Rsp(-) Error Info : = object constraint conflict	LOADED
16	LOADED	Number of Related Object In Use attribute incremented &&its new value=1 Set Related Object In Use flag value to TRUE if any (Start/Stop, Related objects are FI)	IN USE
17	IN USE	Number of Related Object In Use Attribute decremented &&its new value=0 Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill, Related objects are FI)	LOADED

7.2.6.4.3 Push Download State Machine

This kind of transfer uses the following services: Initiate Load, Pull Segment and Terminate Load

A Server may initiate a downloading transfer. In this case it issues an Initiate_Load.Req service primitive to a Client. This primitive tells the Client to initiate downloading transfer for a specified Load Region object. Once the Client has pushed the contents, it issues an Initiate_Load.Rsp(+/-) to the Server.

A Client may initiate a downloading transfer. In this case it issues an Initiate_Load.Req service primitive to a Server. This primitive tells the Server that the contents of a specified load region object will be pushed. Once the Client has pushed the contents, it ends the transfer by issuing a Terminate_Load.req to the Server

7.2.6.4.3.1 Sequencing of Service Primitives

Table 61 summarizes the sequencing of service primitives.

Table 61 – Push Download Sequencing of Service Primitives

FAL User(a Client) Owner of a calling Object		FAL User(a Server) Owner of called Object
Initiate_Load.Ind	←----(T 1)-----	Initiate_Load.Req
Initiate_Load.Req	----->(T2)-----	Initiate_Load.Ind (Transfer started)
Initiate_Load.Cnf	←----(T2)-----	Initiate_Load.Rsp
Push_Segment.Req	----->(T3)-----	Push_Segment.Ind (Transfer continued)
Push_Segment.Cnf	←----(T3)-----	Push_Segment.Rsp
Terminate_Load.Req	----->(T4)-----	Terminate_Load.Ind (Transfer ended)
Terminate_Load.Cnf	←----(T4)-----	Terminate_Load.Rsp
Initiate_Load.Rsp	----->(T1)-----	Initiate_Load.Cnf

7.2.6.4.3.2 Service Parameter Constraints

Table 62 summarizes the service parameter constraints.

Table 62 – Push Download Service Parameter Constraints

Transition Number	Parameter constraints
T1	Initiate_Load: Calling Load Region Key Attribute selected, Load Type : = download, Service To Use : = Push_Segment, Load Service Initiator : = FALSE
T2	Initiate_Load Called Load Region Key Attribute selected, Load Type : = upload Service To Use : = Push_Segment Load Service Initiator : = TRUE
T3	Pull_Segment Load Type : = upload, Segment Number : = ordered 1 to n if any More Follows : = TRUE/FALSE with FALSE for the last segment
T4	Terminate_Load Load Type : = upload, Load Service Used : = Pull_Segment Terminate Reason contained in the response

7.2.6.4.3.3 Download State Transition Table

Table 63 gives the upload state transitions.

Table 63 – Push Download State Table

Transition	Current State	Event	Next State
1	DOWNLOADABLE	Initiate_Load.Ind &&Upload state attribute in state UPLOADABLE Initiate_Load.Rsp(+)	DOWNLOADING
2	DOWNLOADABLE	Initiate_Load.Ind &&Upload state attribute in state <>UPLOADABLE Initiate_Load.Rsp(-) with Error Info : = object constraint conflict	DOWNLOADABLE
3	DOWNLOADING	Push_Segment.Ind &&More Follows=TRUE &&compliance with Inter Segment Req Time Out constraint Push_Segment.Rsp(+)	DOWNLOADING
4	DOWNLOADING	Push_Segment.Ind &&More Follows=FALSE &&compliance with Inter Segment Req Time Out constraint Push_Segment.Rsp(+)	DOWNLOAD SUCCESS
5 exist if supports Inter Segment Req Time Out	DOWNLOADING	Push_Segment.Ind &&More Follows=TRUE II FALSE &&no compliance with Inter Segment Req Time Out constraint Push.Segment.Rsp(-) Discard received data	DOWNLOAD FAILURE
6	DOWNLOADING, DOWNLOAD FAILURE, DOWNLOAD SUCCESS	Abort.Ind Discard received data	DOWNLOADABLE
7	DOWNLOAD FAILURE	Terminate_Load.Ind Terminate_Load.Rsp(+/-) Terminate Reason : = success/failure	DOWNLOADABLE
8	DOWNLOAD SUCCESS	Terminate_Load.Ind &&service fails Terminate_Load.Rsp(-) Terminate Reason : = failure Discard received data	DOWNLOADABLE
9	DOWNLOAD SUCCESS	Terminate_Load.Ind &&service succeeds(e.g. all segments received) Update Contents Size attribute if present Terminate_Load.Rsp(+) Terminate Reason : = success	LOADED
10	LOADED	Initiate_Load.Ind &&Upload State attribute in state UPLOADABLE &&service succeeds(e.g. erasable load region contents) Initiate_Load.Rsp(+)	DOWNLOADING
11	LOADED	Discard.Ind &&Upload State attribute in state UPLOADABLE &&service succeeds(e.g. erasable load region contents) Discard.Rsp(+)	DOWNLOADABLE
12	IN USE	Number of Related object In Use attribute incremented &&its new value > 1 Set Related Object In Use flag value to TRUE if any (Start/Resume)	IN USE

(continued on next page)

Table 63 (concluded)

13	IN USE	Number of Related Object In Use attribute decremented &&its new value >0 Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill)	IN USE
14	LOADED	Initiate_Load.Ind &&Upload State attribute in state <> UPLOADABLE Service fails(e.g. non erasable load region contents) Initiate_Load.Rsp(-) Error Info : = object constraint conflict	LOADED
15	LOADED	Discard.Ind &&Upload State attribute in state <> UPLOADABLE Service fails(e.g. non erasable load region contents) Discard.Rsp(-) Error Info : = object constraint conflict	LOADED
16	LOADED	Number of Related Object In Use attribute incremented &&its new value=1 Set Related Object In Use flag value to TRUE if any (Start/Stop, Related objects are FI)	IN USE
17	IN USE	Number of Related Object In Use Attribute decremented &&its new value=0 Set Related Object In Use flag value to FALSE if any (Reset/Stop/Kill, Related objects are FI)	LOADED

7.2.7 Function Invocation ASE

7.2.7.1 Overview

The FAL function invocation model defines two objects, the stateless action object and the state-oriented function invocation object.

Stateless function invocations, referred to as actions, run to completion when invoked and cannot be interrupted. In addition, some atomic actions return a value in response to being invoked, while others do not. Those that do may be used to model software *functions*, and those that do not may be used to model software *procedures*.

State-oriented function invocations, on the other hand, may be controlled during their execution. Services are defined to suspend, resume, or abort them once they have been invoked. They do not return a value in response to being started. If it is necessary to abort a function invocation once it has been started, then it must be defined as state-oriented.

State-oriented function invocations represent the network view of a specific invocation of a user function. If the user function can be invoked more than once simultaneously, then separate function invocation objects are needed to represent each invocation.

State-oriented function invocations may be used to model software processes or user operations that may be started and controlled.

NOTE An example of a state-oriented function invocation that may be used to model a user operation is the "playback" operation of a video recorder. Once it has been started, the playback operation may be stopped (paused) and later resumed.

To support the concept of software process modeling, the definition of state-oriented functional invocations includes optional attributes that can be used to relate them to load region objects.

The remainder of this clause specifies the class definitions and services for the state-oriented *function invocation* object and the stateless *action* object.

7.2.7.2 Function Invocation Model Class Specifications

This subclause provides formal class definitions for state-oriented function invocation and stateless action objects. Each class definition specifies a type of function invocation as a combination of its attributes and the FAL services defined for it.

7.2.7.2.1 Function Invocation Class Definition

The function invocation class models the state-oriented function invocation. It may be used to model software processes or user functions whose operation may be controlled.

7.2.7.2.1.1 Function Invocation Formal Model**FAL ASE: FUNCTION INVOCATION ASE****CLASS: FUNCTION INVOCATION****CLASS ID: 3****PARENT CLASS: TOP****ATTRIBUTES:**

1	(m)	Attribute:	Access Privilege
1.1	(m)	Attribute:	Password
1.2	(m)	Attribute:	Access Groups
1.3	(m)	Attribute:	Access Rights
2	(m)	Attribute:	Deletable (TRUE,FALSE)
1	(m)	Attribute:	Reusable (TRUE,FALSE) Default : TRUE
2	(m)	Attribute:	Function Invocation State
4	(m)	Attribute:	Number of Related Objects In Use
3	(o)	Attribute:	List of Related Objects
3.1	(o)	Attribute:	ClassID
3.2	(o)	Attribute:	Numeric ID
5.2	(m)	Attribute:	In Use Flag (TRUE, FALSE)
4	(c)	Constraint:	Start Service Supported
4.1	(o)	Attribute:	Start Service Argument Data Type
5	(c)	Constraint:	Stop Service Supported
5.1	(o)	Attribute:	Stop Service Argument Data Type
6	(c)	Constraint:	Resume Service Supported
6.1	(o)	Attribute:	Resume Service Argument Data Type
7	(c)	Constraint:	Reset Service Supported
7.1	(o)	Attribute:	Reset Service Argument Data Type
8	(c)	Constraint:	Kill Service Supported
8.1	(o)	Attribute:	Kill Service Argument Data Type
9	(o)	Attribute:	Last Execution Argument

SERVICES:

1	(o)	OpsService:	Start
2	(o)	OpsService:	Stop
3	(o)	OpsService:	Resume
4	(o)	OpsService:	Reset
5	(o)	OpsService:	Kill

7.2.7.2.1.1.1 Attributes

Access Privilege

This attribute specifies the access controls defined for this function invocation. It is composed of the following:

Password

This attribute contains the password for the access rights. Its value is null if it is not used.

Access Groups

This attribute identifies which of the eight user-defined access groups are defined for the function invocation. More than one access group may be defined.

Access Rights

This attribute defines the type of access defined for the function invocation. Valid values are:

- Right to Delete for Access Groups(see note)
- Right to Start for Access Groups
- Right to Stop for Access Groups
- Right to Resume for Access Groups
- Right to Reset for Access Groups
- Right to Kill for Access Groups
- Right to Delete for the Registered Password (see note)
- Right to Start for the Registered Password
- Right to Stop for the Registered Password
- Right to Resume for the Registered Password
- Right to Reset for the Registered Password
- Right to Kill for the Registered Password
- Right to Delete for all Communication Partners (see note)
- Right to Start for all Communication Partners
- Right to Stop for all Communication Partners
- Right to Resume for all Communication Partners
- Right to Reset for all Communication Partners
- Right to Kill for all Communication Partners

NOTE The right to delete has no meaning if the function invocation object is not deletable. This right requires the use of the Object Management Delete service, and not an operational service of the Function Invocation class.

Deletable

This attribute indicates, when TRUE, that the function invocation can be deleted using the Delete service. The value of this attribute is always TRUE for objects dynamically created with the Object Management ASE Create Service.

Reusable

This attribute indicates, when TRUE, that the function invocation may be executed more than once.

Function Invocation State

This attribute indicates the current state of the function invocation. An enumerated set of values has been defined, and may be extended for local use within the AP to describe transient states of the Function Invocation. These transient state values for this attribute, however, are not visible over the network. The transitions between states and the events that cause transitions are user-defined.

UNRUNNABLE This state indicates that the function invocation is not executing and may not be executed.

IDLE This state indicates that the function invocation is not executing, but is capable of being executed.

RUNNING This state indicates that the function invocation is executing.

STOPPED This state indicates that the execution of a function invocation has been suspended.

STARTING This transient state indicates that the function invocation is being prepared for execution. Function invocations in this state are "ready to run".

STOPPING This transient state indicates that the execution of the function invocation is in the process of being stopped and its context saved.

RESUMING This transient state indicates that the function invocation is in the process of being returned to the executing state from a previously saved context.

RESETTING This transient state indicates that the function invocation is being reset to its initial context and removed from execution.

Number of Related Objects in Use

This attribute indicates the number of related Load Region objects defined for this function invocation that are currently in the IN USE state. This attribute is incremented/decremented each time a related Load Region Object State Attribute goes/leaves the IN USE state. The Load Region State Machine gives the constraints for these transitions.

List of Related Objects

This optional attribute specifies the Load Region objects related to this function invocation.

Class ID

This attribute is the class identifier for the related object.

Numeric ID

This attribute is the numeric identifier for the related object.

In-use flag

This attribute indicates, when TRUE, that the related object is currently in the IN USE state. This attribute is updated each time the related Load Region Object state attribute goes/leaves the IN USE state. The Load Region State Machine gives the constraints for these transitions

Start Service Argument Data Type

This attribute identifies the data type for each of the execution arguments for the start service supported by this function invocation. This attribute is present if the function invocation supports the start service. If the start service is supported but has no execution argument, the value of this attribute is NULL.

Stop Service Argument Data Type

This attribute identifies the data type for the execution argument for the stop service supported by this function invocation. This attribute is present if the function invocation supports the stop service. If the stop service is supported but has no execution argument, the value of this attribute is NULL.

Resume Service Argument Data Type

This attribute identifies the data type for the execution argument for the resume service supported by this function invocation. This attribute is present if the function invocation supports the resume service. If the resume service is supported but has no execution argument, the value of this attribute is NULL.

Reset Service Argument Data Type

This attribute identifies the data type for the execution argument for the reset service supported by this function invocation. This attribute is present if the function invocation supports the reset service. If the reset service is supported but has no execution argument, the value of this attribute is NULL.

Kill Service Argument Data Type

This attribute identifies the data type for the execution argument for the kill service supported by this function invocation. This attribute is present if the function invocation supports the kill service. If the kill service is supported but has no execution argument, the value of this attribute is NULL.

Last Execution Argument

This optional attribute contains the value for the Execution Argument that was last sent by the Client.

7.2.7.2.1.1.2 Services

Start

This optional service is used to request the function invocation to start executing. If this service is not present, it is assumed that the function invocation will be started using local means.

Stop

This optional service is used to request the function invocation to stop executing.

Resume

This optional service is used to request the function invocation to continue its execution.

Reset

This optional service is used to request the function invocation to return to its initial context and state.

Kill

This optional service is used to request that the current execution of the function invocation be aborted so that it may not be resumed. Once killed, the context of the function execution is lost.

7.2.7.2.2 Action Class Definition

The Action Class is used to define stateless function invocations. Stateless function invocations run to completion when invoked. Their execution may not be aborted using the services of the FAL. They may return a result using the Return service.

7.2.7.2.2.1 Action Formal Model

FAL ASE: FUNCTION INVOCATION ASE

CLASS: ACTION

CLASS ID: 19

PARENT CLASS: TOP

ATTRIBUTES:

1	(m)	Attribute:	Reusable (TRUE,FALSE) Default : TRUE
2	(o)	Attribute:	List of Related Objects
3.1	(m)	Attribute:	ClassID
3.2	(m)	Attribute:	Numeric ID
3.2	(m)	Attribute:	In Use Flag (TRUE, FALSE)
4	(o)	Attribute:	Action Invoke Service Argument Data Type
5	(c)	Constraint:	Action Return Service Supported
5.1	(o)	Attribute:	Action Return Service Argument Data Type

SERVICES:

1	(m)	OpsService:	Action Invoke
2	(o)	OpsService:	Action Return

7.2.7.2.2.1.1 Attributes

Reusable

This attribute indicates, when TRUE, that the action may be executed more than once.

List of Related Objects

This optional attribute specifies the Load Region objects related to this function invocation.

Class ID

This attribute is the class identifier for the related object.

Numeric ID

This attribute is the numeric identifier for the related object.

In-use Flag

This attribute indicates, when TRUE, that the related object is currently in the IN USE state. This attribute is updated each time the Load Region Object State attribute goes/leaves the IN USE state. The Load Region State Machine gives the constraints for these transitions.

Action Invoke Service Argument Data Type

This attribute identifies the data type for the execution argument for the action invoke service supported by this action. This attribute is present if the action supports the action invoke service. If the action invoke service is supported but has no execution argument, the value of this attribute is NULL.

Action Return Service Argument Data Type

This attribute identifies the data type for the execution argument for the action invoke service supported by this action. This attribute is present if the action supports the action invoke service. If the action invoke service is supported but has no execution argument, the value of this attribute is NULL.

7.2.7.2.1.2 Services

Action Invoke

This mandatory service is used to call the action object with an argument list.

Action Return

This optional service is used by the action object to return the results of its invocation.

7.2.7.3 Function Invocation Model Service Specifications

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

Start

Stop

Resume

Reset

Kill

Action Invoke

Action Return

7.2.7.3.1 Start Service

This confirmed service is used to request that a function invocation be started.

7.2.7.3.1.1 Service Primitives

The service parameters for this service are shown in Table 64.

Table 64 – Start Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Key Attribute	M	M(=)		
Execution Argument	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)
Function Invocation State			C	C(=)

Argument

This parameter carries the parameters of the service invocation.

Key attribute

This parameter specifies one of the key attributes of the function invocation object.

Execution Argument

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data Types for the function invocation object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Function Invocation State

This conditional parameter contains the current state of the function invocation. It is present if the error code indicates an object state conflict.

7.2.7.3.1.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.7.3.2 Stop Service

This confirmed service is used to stop a function invocation retaining its context so that it may be resumed.

7.2.7.3.2.1 Service Primitives

The service parameters for this service are shown in Table 65.

Table 65 – Stop Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Key Attribute	M	M(=)		
Execution Argument	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)
Function Invocation State			C	C(=)

Argument

This parameter carries the parameters of the service invocation.

Key Attribute

This parameter specifies one of the key attributes of the function invocation object.

Execution Argument

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data Types for the function invocation object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Function Invocation State

This conditional parameter contains the current state of the function invocation. It is present if the error code indicates an object state conflict.

7.2.7.3.2.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.7.3.3 Resume Service

This confirmed service is used to resume the execution of a function invocation that has been stopped. Execution is resumed using the context saved when the function invocation was stopped.

7.2.7.3.3.1 Service Primitives

The service parameters for this service are shown in Table 66.

Table 66 – Resume Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Key Attribute	M	M(=)		
Execution Argument	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)
Function Invocation State			C	C(=)

Argument

This parameter carries the parameters of the service invocation.

Key Attribute

This parameter specifies one of the key attributes of the function invocation object.

Execution Argument

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data Types for the function invocation object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Function Invocation State

This conditional parameter contains the current state of the function invocation. It is present if the error code indicates an object state conflict.

7.2.7.3.3.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.7.3.4 Reset Service

This confirmed service is used to reset a function invocation with its initial context. Its initial context is defined as the context of the function invocation set by its initialization procedures.

7.2.7.3.4.1 Service Primitives

The service parameters for this service are shown in Table 67.

Table 67 – Reset service parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Key Attribute	M	M(=)		
Execution Argument	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)
Function Invocation State			C	C(=)

Argument

This parameter carries the parameters of the service invocation.

Invoke ID

This parameter identifies this invocation of the service. It is used to associate service requests with responses. Therefore, no two outstanding service invocations may be identified by the same Invoke ID value.

Key Attribute

This parameter specifies one of the key attributes of the function invocation object.

Execution Argument

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data Types for the function invocation object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Function Invocation State

This conditional parameter contains the current state of the function invocation. It is present if the error code indicates an object state conflict.

7.2.7.3.4.2 Service procedure

The Confirmed Service Procedure Specified In Clause 4 Applies To This Service.

7.2.7.3.5 Kill Service

This confirmed service is used to abort the execution of a function invocation so that it may not be restarted or resumed.

7.2.7.3.5.1 Service Primitives

The service parameters for this service are shown in Table 68.

Table 68 – Kill service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Key Attribute	M	M(=)		
Execution Argument	U	U(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)
Function Invocation State			C	C(=)

Argument

This parameter carries the parameters of the service invocation.

Key Attribute

This parameter specifies one of the key attributes of the function invocation object.

Execution Argument

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data Types for the function invocation object.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Function Invocation State

This conditional parameter contains the current state of the function invocation. It is present if the error code indicates an object state conflict.

7.2.7.3.5.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

7.2.7.3.6 Action Invoke Service

This unconfirmed service is used to invoke the execution of an action object.

7.2.7.3.6.1 Service Primitives

The service parameters for this service are shown in Table 69.

Table 69 – Action Invoke Service Parameters

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Destination DL-Address	C	
Source DL-Address		C
Key Attribute	M	M(=)
Action Invocation ID	M	M(=)
Execution Argument	U	U(=)
Timeliness		C
Duplicate FAL PDU Body		C

Argument

This parameter carries the parameters of the service invocation.

Destination DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the destination address to which the requested Action Invoke is to be sent.

Source DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the source address from which the indicated Action Invoke is to be sent.

Key Attribute

This parameter specifies one of the key attributes of the action object.

Action Invocation ID

This parameter identifies a specific invocation of the action object. It permits correlation of the invocation and its result as reported with a Return service.

Execution Argument

This optional parameter carries the execution argument. This parameter is present when its data type is defined in the List of Service Argument Data Types for the action object.

Timeliness

This conditional parameter indicates the Data Link Layer timeliness status for the transfer of this service request. This parameter is present if it is supported on the specified AP.

Duplicate FAL PDU Body

This conditional parameter indicates whether or not the receipt of a duplicate FAL PDU has been detected by the Data Link Layer. It is present if supported in the DL Mapping Attributes of the AREP.

7.2.7.3.6.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

7.2.7.3.7 Action Return Service

This unconfirmed service is used return the results from an invocation of an action object.

7.2.7.3.7.1 Service Primitives

The service parameters for this service are shown in Table 70.

Table 70 – Action Return Service Parameters

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Destination DL-Address	C	
Source DL-Address		C
Action Invocation ID	M	M(=)
Action Success	S	S(=)
Return Value	U	U(=)
Action Failure	S	S(=)
Error Info	M	M(=)

Argument

This parameter carries the parameters of the service invocation.

Destination DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the destination address to which the requested Action Return is to be sent.

Source DL-Address

This parameter exists only when the corresponding AREP supports it and is configured with a Remote Address Configuration Type of FREE. It indicates the source address from which the indicated Action Return is to be sent.

Action Invocation ID

This parameter identifies the specific invocation of the action object whose results are being returned.

Action Success

This selection type parameter is present if the action was successful.

Return Value

This optional parameter carries the results of this invocation of the action.

Action Failure

This selection type argument is present if the action was not successful.

7.2.7.3.7.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

The AP containing the action submits an Action Return Service request primitive to return the results of a previous invocation of the action. An action invocation identifier is included in the request to identify the particular invocation. This identifier is used only by the receiving user, and is only conveyed by the FAL.

7.2.7.4 Function Invocation State Machine

Table 71 gives the State Transitions for a Function Invocation object. The state “NON EXISTENT” indicates that the object does not exist. It is not defined as a valid value for the function invocation state attribute because attributes are created and deleted with the object (and therefore have no value when the object does not exist).

Table 71 – State Transitions For A Function Invocation Object

Transition	Current State	Events	Next State
1	NON EXISTENT	Create.Ind && List of Attribute IDs and Initial Values parameter = List of Load Region Object IDs && Specified Load Region objects available For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Create.Rsp(+)	Idle
2	NON EXISTENT	Create.Ind && List of Attribute IDs and Initial Values parameter <> List of Load Region Object IDs Specified Load Region objects not available Create.Rsp(-)	NON EXISTENT
3	IDLE	Delete.Ind && Deleteable=TRUE Delete.Rsp(+)	NON EXISTENT
4	IDLE	Delete.Ind && Deleteable=FALSE Delete.Rsp(-)	IDLE
5	IDLE	Start.Ind	STARTING
6	STARTING	Start succeeds, e.g. Related Load Region Object in state LOADED or IN-USE For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Start.Rsp(+)	RUNNING
7	STARTING	Start fails, non destructive: e.g. Related Load Region Object not in state LOADED or IN USE Start.Rsp(-)	IDLE
8	STARTING	Start fails, destructive Start.Rsp(-)	UN-RUNNABLE
9	RUNNING	Stop.Ind	STOPPING
10	STOPPING	Stop succeeds: e.g. Related Load Region Object in state IN USE For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Stop.Rsp(+)	STOPPED

(continued on next page)

11	STOPPING	Stop Execution fails, non destructive Stop.Rsp(-)	RUNNING
12	STOPPING	Stop fails, destructive For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Stop.Rsp(-)	UN-RUNNABLE
13	STOPPED	Resume.Ind	RESUMING
14	RESUMING	Resume succeeds: e.g. Related Load Region Object in state Loaded or In-Use For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Resume.Rsp(+)	RUNNING
15	RESUMING	Resume fails, non destructive: e.g. Load Region not in state LOADED or IN-USE Resume.Rsp(-)	STOPPED
16	RESUMING	Resume fails, destructive For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Resume.Rsp(-)	UNRUNNABLE
17	STOPPED	Reset.Ind	RESETTING
18	RESETTING	Reset succeeds, reusable=TRUE Reset.Rsp(+)	IDLE
19	RESETTING	Reset succeeds, reusable = false Reset.Rsp(+)	UNRUNNABLE
20	RESETTING	Reset fails, non destructive Reset.Rsp(-)	STOPPED
21	RESETTING	Reset fails, destructive Reset.Rsp(-)	UNRUNNABLE
22	RESETTING	Kill.Ind Kill.Rsp(+)	UNRUNNABLE
23	STOPPING	Kill.Ind For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Kill.Rsp(+)	UNRUNNABLE

(continued on next page)

24	RUNNING	End of Program && Reusable=TRUE For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported)	IDLE
25	RUNNING	End of Program && Reusable = false For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported)	UNRUNNABLE
26	RUNNING	Program stopped For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported)	STOPPED
27	STOPPED	Delete.Ind && this Function Invocation is not deletable Delete.Rsp(-)	STOPPED
28	STOPPED	Kill.Ind Kill.Rsp(+)	UNRUNNABLE
29	STOPPED	Delete.Ind && this Function Invocation is deletable Delete.Rsp(+)	NON EXISTENT
30	RESUMING	Kill.Ind Kill.Rsp(+)	UNRUNNABLE
31	RUNNING	Kill.Ind For the Function Invocation object, update the attributes: Number of Related Object In Use List of Related Objects(if supported) For each related Load Region Object, update the attributes: Number of Related Objects In Use List of Related Objects(if supported) Kill.Rsp(+)	UNRUNNABLE
32	STARTING	Kill.Ind Kill.Rsp(+)	UNRUNNABLE
33	IDLE	Kill.Ind Kill.Rsp(+)	UNRUNNABLE
34	UNRUNNABLE	Delete.Ind Delete.Rsp(+)	NON EXISTENT

7.3 Type 1 ARs

7.3.1 Queued User-triggered Uni-directional (QUU) AR Endpoint Formal Model

7.3.1.1 Description

This class is defined to support the on-demand queued distribution of unconfirmed services to one or more application processes. The behavior of this type of AR can be described as follows.

An AR ASE user wishing to convey a request APDU submits an AR ASE Service Data Unit to the sending endpoint of the AR. The AREP sending the request APDU submits it to its underlying layer for transfer. The underlying layer sends it at its next opportunity. The AREP receiving the request APDU from its underlying layer delivers it to the AR ASE user in the order that it was received.

The following summarizes the characteristics of this AREP class.

Roles: REPORT SOURCE
 REPORT SINK
Cardinality: 1-to-n
Timeliness: No

7.3.1.2 Formal Class Definition

FAL ASE: **AR ASE**
CLASS: Queued User-triggered Uni-directional AR Endpoint
CLASS ID: 36
PARENT CLASS: AR Endpoint
NETWORK MANAGEMENT ATTRIBUTES:
1. (m) Attribute: Role (REPORT SOURCE, REPORT SINK)
2. (m) Attribute: AREP State (OPEN, CLOSED)
3. (m) Attribute: Remote Address Configuration Type (FREE, LINKED)
4. (m) Attribute: DL Mapping Reference
SERVICES:
1. (o) OpsService: Unconfirmed Send

7.3.1.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

REPORT SOURCEEndpoints of this type distribute reports to multiple endpoints using unconfirmed services operating over connectionless data link services.

REPORT SINKEndpoints of this type receive reports from source endpoints using unconfirmed services operating over connectionless data link services.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

Remote Address Configuration Type

This attribute specifies how the remote address of the AREP is configured.

FREE	The value of “FREE” indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically in the Unconfirmed service request.
LINKED	The value of “LINKED” indicates that the destination of the FAL-PDUs is configured with the RemoteDisapAddress attribute contained in the DL Mapping (specified in IEC 61158-6).

DL Mapping Reference

This attribute is a reference to the underlying Data Link Layer mapping. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

7.3.1.2.2 Services**Unconfirmed Send**

This optional service is used to send an unconfirmed service on an AR.

7.3.2 Queued User-triggered Bi-directional Connection-Oriented (QUB-Co) AR Endpoint Formal Model**7.3.2.1 Description**

This class is defined to support the on-demand exchange of confirmed services between two application processes. Unconfirmed services are not supported by this type of AR. It uses connection-oriented data link services for the exchanges. The behavior of these classes is described as follows.

An AR ASE user wishing to convey a request APDU submits it as an AR ASE Service Data Unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity.

The AREP receiving the request APDU from its underlying layer, queues it for delivery to its AR ASE user in the order that it was received.

For a confirmed service request the AREP receiving the request APDU accepts the corresponding response APDU from its AR ASE user and queues it to the underlying layer for transfer.

The AREP that issued the request APDU receives the response APDU from its underlying layer and queues it for delivery to its AR ASE user in the order that it was received. It also stops its associated service response timer.

The following summarizes the characteristics of this AREP class.

Roles:	Client
	Server
	Peer
Cardinality:	1-to-1
Timeliness:	No

7.3.2.2 Formal Class Definition

FAL ASE: **AR ASE**

CLASS: Queued User-triggered Bi-directional Connection-Oriented AREP

CLASS ID: 34

PARENT CLASS: AR Endpoint

NETWORK MANAGEMENT ATTRIBUTES:

1. (m) Attribute: Role (CLIENT, SERVER, PEER)
2. (m) Attribute: AREP State
3. (m) Attribute: Remote Address Configuration Type (FREE, LINKED)
4. (m) Attribute: Maximum Outstanding Requests Calling
5. (m) Attribute: Maximum Outstanding Requests Called
6. (m) Attribute: Initiator (TRUE, FALSE)
7. (o) Attribute: Remote AP Name
8. (m) Attribute: Transmit DL Mapping Reference
9. (m) Attribute: Receive DL Mapping Reference

SERVICES:

1. (o) OpsService: Confirmed Send
2. (o) OpsService: Establish

7.3.2.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

PEER (CLIENT/SERVER) Endpoints of this type may perform as either clients or servers, or both simultaneously. Endpoints of this type must indicate whether or not they are to be the initiator of the AR establishment process.

CLIENT Endpoints of this type issue confirmed service Request-APDUs to servers and receive confirmed service Response-APDUs.

SERVER Endpoints of this type receive confirmed and unconfirmed service Request-APDUs from clients and issue confirmed service Response-APDUs to them.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

Remote Address Configuration Type

This attribute specifies how the remote address of the AREP is configured. The valid values are:

FREE	The value of "FREE" indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically in the Unconfirmed service request.
LINKED	The value of "LINKED" indicates that the destination of the FAL-PDUs is configured with the RemoteDisapAddress attribute contained in the DL Mapping (specified in IEC 61158-6).

Max Outstanding Requests Calling

This conditional attribute indicates the maximum number of responses that the AREP may be expecting from the remote AREP.

Max Outstanding Requests Called

This conditional attribute indicates the maximum number of responses that the AR Endpoint may be expecting from its local user.

Initiator

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR must be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

Remote AP Name

This optional attribute specifies the name of the AP attached at the remote AREP.

Transmit DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the transmit conveyance path for this AREP. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

Receive DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the receive conveyance path for this AREP. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

7.3.2.2.2 Services**Confirmed Send**

This optional service is used to send a confirmed service on an AR.

Establish

This service is used to establish an AR.

7.3.3 Queued User-triggered Bi-directional Connectionless (QUB-CL) AR Endpoint Formal Model**7.3.3.1 Description**

This class is defined to support the on-demand exchange of confirmed and unconfirmed services between two or more application processes. This class uses connectionless data link services for the exchanges. The Data Link Layer transfers may be either acknowledged or unacknowledged. The behavior of this class is described as follows.

An AR ASE user wishing to convey a request APDU submits it as an AR ASE Service Data Unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity.

The AREP receiving the request APDU from its underlying layer, queues it for delivery to its AR ASE user in the order that it was received.

For a confirmed service request the AREP receiving the request APDU accepts the corresponding response APDU from its AR ASE user and queues it to the underlying layer for transfer.

The AREP that issued the request APDU receives the response APDU from its underlying layer and queues it for delivery to its AR ASE user in the order that it was received. It also stops its associated service response timer.

The following summarizes the characteristics of this AREP class.

Roles: Client
Server
Peer
Cardinality: 1-to-n
Timeliness: No

7.3.3.2 Formal Class Definition

FAL ASE: **AR ASE**
CLASS: Queued User-triggered Bi-directional Connectionless AREP
CLASS ID: 45
PARENT CLASS: AR Endpoint

NETWORK MANAGEMENT ATTRIBUTES:

1	(m)	Attribute:	Role (CLIENT, SERVER, PEER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	Remote Address Configuration Type (FREE, LINKED)
4	(m)	Attribute:	Maximum Outstanding Requests Calling
5	(m)	Attribute:	Maximum Outstanding Requests Called
6	(m)	Attribute:	Initiator (TRUE, FALSE)
7	(o)	Attribute:	Remote AP Name
8	(m)	Attribute:	Transmit DL Mapping Reference
9	(m)	Attribute:	Receive DL Mapping Reference

SERVICES:

1	(o)	OpService:	Confirmed Send
2	(o)	OpService:	Unconfirmed Send
3	(o)	OpService:	Establish
4	(o)	OpService:	Abort

7.3.3.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

PEER (CLIENT/SERVER) Endpoints of this type may perform as either clients or servers, or both simultaneously. Endpoints of this type must indicate whether or not they are to be the initiator of the AR establishment process.

CLIENT Endpoints of this type issue confirmed and unconfirmed service Request-APDUs to servers and receive confirmed service Response-APDUs.

SERVER Endpoints of this type receive confirmed and unconfirmed service Request-APDUs from clients and issue confirmed service Response-APDUs to them. They may also issue unconfirmed service Request APDUs to clients.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

Remote Address Configuration Type

This attribute specifies how the remote address of the AREP is configured. The valid values are:

FREE	The value of "FREE" indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically.
LINKED	The value of "LINKED" indicates that the destination of the FAL-PDUs is configured with the RemoteDisapAddress attribute contained in the DL Mapping (specified in IEC 61158-6).

Max Outstanding Requests Calling

This conditional attribute indicates the maximum number of responses that the AREP may be expecting from the remote AREP.

Max Outstanding Requests Called

This conditional attribute indicates the maximum number of responses that the AR Endpoint may be expecting from its local user.

Initiator

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR must be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

Remote AP Name

This optional attribute specifies the name of the AP attached at the remote AREP.

Transmit DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the transmit conveyance path for this AREP. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

Receive DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the receive conveyance path for this AREP. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

7.3.3.2.2 Services

Confirmed Send

This optional service is used to send a confirmed service on an AR.

Unconfirmed Send

This optional service is used to send an unconfirmed service on an AR.

Establish

This service is used to establish an AR.

Abort

This service is used to abort an AR.

7.3.4 Queued User-triggered Bi-directional with Flow Control (QUB-FC) AR Endpoint Formal Model

7.3.4.1 Description

This class is defined to support asynchronous operation of the client/server model using queues. It provides for the conveyance of confirmed and unconfirmed services using application layer flow control for unconfirmed services and uses connection-oriented data link services for the exchanges. The data link priority for the transfers is specified separately for each transfer.

The behavior of this class is described as follows.

An AR ASE user wishing to convey a request APDU submits it as an AR ASE Service Data Unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity with exception of unconfirmed acknowledged services. The latter will be ignored if the outstanding service counter has reached the negotiated maximum.

The AREP receiving the request APDU from its underlying layer, queues it for delivery to its AR ASE user, in the order that it was received. If the received APDU contains an unconfirmed acknowledged request, the AREP issues the appropriated indication to the AR ASE user and a UCA_AckPDU to the underlying layer for transfer.

For a confirmed service request the AREP receiving the request APDU accepts the corresponding response APDU from its AR ASE user and queues it to the underlying layer for transfer.

The AREP that issued the request APDU receives the response APDU from its underlying layer and queues it for delivery to its AR ASE user in the order that it was received and decrements the appropriate flow control counter.

Additionally the AREP monitors the connection in cases if no service from the AR ASE user is outstanding by sending an IdlePDU to the peer instance. The TCTimer (Transmit Client Timer) and the TSTimer (Transmit Server Timer) monitor the sending of IdlePDUs on both connection endpoints. The RCTimer (Receive Client Timer) and RSTimer (Receive Server Timer) monitor the liveness of the connection on both connection endpoints. The RCTimer or RSTimer expire if no PDU (also no IdlePDU) has been received during a certain amount of time and causes the termination of the AREP connection.

Furthermore, the AR ASE user at server site is able to interrupt or continue the data flow by issuing the AR-XON-XOFF service.

The following summarizes the characteristics of this AREP class.

Roles	Client
	Server
	Peer
Cardinality	1-to-1
Timeliness	No

7.3.4.2 Formal Class Definition

FAL ASE:		AR ASE
CLASS:		Queued User-triggered Bi-directional with Flow Control AREP
CLASS ID:		46
PARENT CLASS:		AR Endpoint
NETWORK MANAGEMENT ATTRIBUTES:		
1	(m) Attribute:	Role (CLIENT, SERVER, PEER)
2	(m) Attribute:	AREP State
3	(m) Attribute:	Remote Address Configuration Type (FREE, LINKED)
4	(m) Attribute:	Maximum Outstanding Requests Calling
5	(m) Attribute:	Maximum Outstanding Requests Called
6	(m) Attribute:	Max Outstanding Unconfirmed Requests Client (maxUCSC)
7	(m) Attribute:	Max Outstanding Unconfirmed Requests Server (maxUCSS)
8	(m) Attribute:	Maximum Outstanding Requests Client (maxOSCC)
9	(m) Attribute:	Maximum Outstanding Requests Server (maxOSCS)
10	(m) Attribute:	Initiator (TRUE, FALSE)
11	(o) Attribute:	Remote AP Name
12	(m) Attribute:	Transmit DL Mapping Reference
13	(m) Attribute:	Receive DL Mapping Reference
14	(m) Attribute:	CIU
SERVICES:		
1	(o) OpsService:	Confirmed Send
2	(o) OpsService:	Unconfirmed Send
3	(o) OpsService:	Establish
4	(o) OpsService:	Abort

7.3.4.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

PEER (CLIENT/SERVER) Endpoints of this type may perform as either clients or servers, or both simultaneously. Endpoints of this type must indicate whether or not they are to be the initiator of the AR establishment process.

CLIENT Endpoints of this type issue confirmed and unconfirmed service Request-APDUs to servers and receive confirmed service Response-APDUs.

SERVER Endpoints of this type receive confirmed and unconfirmed service Request-APDUs from clients and issue confirmed service Response-APDUs to them. They may also issue unconfirmed service Request APDUs to clients.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

Remote Address Configuration Type

This attribute specifies how the remote address of the AREP is configured. The valid values are:

FREE	The value of “FREE” indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically in the Unconfirmed service request.
LINKED	The value of “LINKED” indicates that the destination of the FAL-PDUs is configured with the RemoteDlsapAddress attribute contained in the DL Mapping (specified in IEC 61158-6).

Max outstanding Requests Calling

This attribute indicates the maximum number of responses that the AREP may be expecting from the remote AREP.

Max Outstanding Requests Called

This attribute indicates the maximum number of responses that the AR Endpoint may be expecting from its local user.

Max Outstanding Unconfirmed Requests Client (maxUCSC)

This attribute indicates the maximum number of requests that the AR Endpoint may be expecting from its local user.

Max Outstanding Unconfirmed Requests Server (maxUCSS)

This attribute indicates the maximum number of requests that the AREP may be expecting from the remote AREP.

Maximum Outstanding Requests Client (maxOSCC)

This attribute indicates the maximum number of responses that the AREP may be expecting from the remote AREP.

Maximum Outstanding Requests Server (maxOSCS)

This attribute indicates the maximum number of responses that the AR Endpoint may be expecting from its local user.

Initiator

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR must be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

Remote AP Name

This optional attribute specifies the name of the AP attached at the remote AREP.

Transmit DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the transmit conveyance path for this AREP. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

Receive DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the receive conveyance path for this AREP. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

CIU

This attribute (CIU - Control Interval User-triggered) contains the control interval for monitoring of a user triggered connection. The attribute is set by network management and used by the ARPM. If this attribute contains a value not 0 then the ARPM transmits an Idle PDU while AR ASE user has not passed any service during the control interval (CIU/3). If CIU expires then the connections will be terminated.

7.3.4.2.2 Services**Confirmed Send**

This optional service is used to send a confirmed service on an AR.

Establish

This service is used to establish an AR.

Abort

This service is used to disconnect an AR.

Unconfirmed Send

This service is used to send an unconfirmed service on an AR.

7.3.5 Queued User-Triggered Bi-directional with Segmentation (QUB-Seg) AR Endpoint Formal Model**7.3.5.1 Description**

This class is defined to support the on-demand exchange of confirmed and unconfirmed services between two application processes. It uses connection-oriented datalink services for the exchanges. It supports segmentation of APDUs within the Application Layer. The data link priority for the transfers is specified separately for each transfer direction.

An AR ASE service data unit (if its size is greater than a single transferable element) can be split into several APDUs which are sent to its underlying layer to be transferred. The underlying layer sends each segment at the next opportunity. The AREP receiving the request APDU segments from its underlying layer reassembles the APDU and delivers it to the AR ASE. Delivery of multiple APDUs is done in the order in which they are completely received.

The following summarizes the characteristics of this AREP class.

Roles:	Client
	Server
	Peer
Cardinality:	1-to-1
Timeliness:	No

7.3.5.2 Formal Class Definition

FAL ASE: AR ASE
CLASS: Queued User-triggered Bi-directional Segmentation AREP
CLASS ID: 45
PARENT CLASS: AR Endpoint

NETWORK MANAGEMENT ATTRIBUTES:

1. (m) Attribute: Role (CLIENT, SERVER, PEER)
2. (m) Attribute: AREP State
3. (m) Attribute: Remote Address Configuration Type (FREE, LINKED)
4. (m) Attribute: Maximum Outstanding Requests Calling
5. (m) Attribute: Maximum Outstanding Requests Called
6. (m) Attribute: Initiator (TRUE, FALSE)
7. (o) Attribute: Remote AP Name
8. (m) Attribute: Transmit DL Mapping Reference
9. (m) Attribute: Receive DL Mapping Reference

SERVICES:

1. (o) OpsService: Confirmed Send
2. (o) OpsService: Unconfirmed Send
3. (o) OpsService: Establish
4. (o) OpsService: Reject
5. (o) OpsService: Abort

7.3.5.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

PEER (CLIENT/SERVER) Endpoints of this type may perform as either clients or servers, or both simultaneously. Endpoints of this type must indicate whether or not they are to be the initiator of the AR establishment process.

CLIENT Endpoints of this type issue confirmed service Request-APDUs to servers and receive confirmed service Response-APDUs and unconfirmed service Request PDU.

SERVER Endpoints of this type receive confirmed and unconfirmed service Request-APDUs from clients and issue confirmed service Response-APDUs and unconfirmed service Request PDU to them. They may also issue unconfirmed service Request APDUs to clients.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

Remote Address Configuration Type

This attribute specifies how the remote address of the AREP is configured. The valid values are:

FREE	The value of "FREE" indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically.
LINKED	The value of "LINKED" indicates that the destination of the FAL-PDUs is configured with the RemoteDisapAddress attribute contained in the DL Mapping (specified in IEC 61158-6).

Max Outstanding Requests Calling

This conditional attribute indicates the maximum number of responses that the AREP may be expecting from the remote AREP.

Max Outstanding Requests Called

This conditional attribute indicates the maximum number of responses that the AR Endpoint may be expecting from its local user.

Initiator

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR must be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

Remote AP Name

This optional attribute specifies the name of the AP attached at the remote AREP.

Transmit DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the transmit conveyance path for this AREP. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

Receive DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the receive conveyance path for this AREP. DL mappings for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

7.3.5.2.2 Services**Confirmed Send**

This optional service is used to send a confirmed service on an AR.

Unconfirmed Send

This optional service is used to send an unconfirmed service on an AR.

Establish

This service is used to establish an AR.

Abort

This service is used to Abort an AR.

Reject

This provider initiated service is used to signal the detection of a protocol error.

7.3.6 Buffered User-triggered Bi-directional (BUB) AR Endpoint Formal Model

7.3.6.1 Description

This class is defined to support the on-demand exchange of confirmed and of unconfirmed services between two application processes using buffers.

The behavior of this type of AR can be described as follows.

An AR ASE user wishing to convey a request or response APDU submits it to its AREP as an AR ASE Service Data Unit. The AREP sending the request or response APDU writes it into the Data Link Layer buffer, completely replacing the existing contents of the buffer. The Data Link Layer transfers the buffer contents at the earliest transfer opportunity. The buffer may also be transferred whenever a request to transmit (AR Compel) is received, either locally or from the network.

The AREP receiving the request or response APDU from its underlying layer, buffers it for delivery. If the APDU is a response, the endpoint stops its associated timer.

If the AREP that sent the APDU receives another APDU before the buffer contents are transmitted, the buffer contents will be replaced with the new APDU, and the previous APDU will be lost. When the buffer contents are transmitted, the AR ASE notifies the user of the transmission.

At the receiving endpoint, the APDU is received from the network and written into the buffer, completely replacing the existing contents of the buffer. The endpoint notifies the user that the APDU has arrived and delivers it to the user according to the local user interface. If the APDU has not been delivered before the next APDU arrives, it will be replaced by the next APDU and lost.

An FAL user receiving the buffered transmission may later request to receive the currently buffered APDU.

The following summarizes the characteristics of this AREP class.

Roles:	Client Server Peer
Cardinality:	1-to-1
Timeliness:	No

7.3.6.2 Formal Class Definition

FAL ASE:	AR ASE
CLASS:	Buffered User-triggered Bi-directional AREP
CLASS ID:	42
PARENT CLASS:	AR Endpoint

NETWORK MANAGEMENT ATTRIBUTES:

1	(m)	Attribute:	Role (CLIENT, SERVER, PEER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	Confirmed Services Flag(TRUE/FALSE)
4	(m)	Attribute:	Initiator (TRUE, FALSE)
5	(m)	Attribute:	Transmit DL Mapping Reference
6	(m)	Attribute:	Receive DL Mapping Reference
7	(c)	Constraint:	Confirmed Services = TRUE
7.1	(m)	Attribute:	Max Outstanding Requests Calling
7.2	(m)	Attribute:	Max Outstanding Requests Called
8	(m)	Attribute:	Remote AP Name

SERVICES:

1	(o)	OpsService:	Unconfirmed Send
2	(o)	OpsService:	AR Compel
3	(o)	OpsService:	Confirmed Send
4	(o)	OpsService:	Establish
5	(o)	OpsService:	Get Buffered Message

7.3.6.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

PEER (CLIENT/SERVER) Endpoints of this type may perform as either clients or servers, or both simultaneously. Endpoints of this type must indicate whether or not they are to be the initiator of the AR establishment process.

CLIENT Endpoints of this type issue confirmed and unconfirmed service Request-APDUs to servers and receive confirmed service Response-APDUs.

SERVER Endpoints of this type receive confirmed and unconfirmed service Request-APDUs from clients and issue confirmed service Response-APDUs to them. They may also issue unconfirmed service Request APDUs to clients.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

Confirmed Services Flag

This attribute specifies, when TRUE, that this endpoint supports the conveyance of confirmed service requests.

Initiator

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR must be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

Transmit DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the transmit conveyance path for this AREP. DL-Mapping References for AREPs are specified in IEC 61158-6.

Receive DL Mapping Reference

This attribute provides the mapping to the underlying layer for the receive conveyance path for this AREP. DL-Mapping References for AREPs are specified in IEC 61158-6.

Max Outstanding Requests Calling

This conditional attribute indicates the maximum number of responses that the AREP may be expecting from the remote AREP. This attribute is present only if this endpoint supports confirmed services (confirmed services = TRUE).

Max Outstanding Requests Called

This conditional attribute indicates the maximum number of responses that the AR Endpoint may be expecting from its local user. This attribute is present only if this endpoint supports confirmed services (Confirmed Services = TRUE).

Remote AP Name

This attribute specifies the AP attached to the remote AREP.

7.3.6.2.2 Services

Unconfirmed Send

This optional service is used to send an unconfirmed service on an AR.

AR Compel

This optional service is used to cause an APDU locally or remotely buffered for transmission in the Data Link Layer to be transmitted at the next available opportunity.

Confirmed Send

This optional service is used to send a confirmed service on an AR.

Establish

This service is used to establish an AR.

Get Buffered Message

This local service is used to retrieve an APDU from the buffer used by an AR.

7.3.7 Buffered Network-Scheduled Uni-directional (BNU) AR Endpoint Formal Model

7.3.7.1 Description

This class is defined to support the “push” model for scheduled, buffered distribution of unconfirmed services to one or more application processes.

The behavior of this type of AR can be described as follows.

An AR ASE user wishing to convey a request or response APDU submits it as an AR ASE Service Data Unit to its AREP for distribution. The AREP sending the request or response APDU writes it into the Data Link Layer buffer, completely replacing the existing contents of the buffer. The Data Link Layer transfers the buffer contents at the next scheduled transfer opportunity. The buffer is also transferred whenever a request to transmit (AR Compel) is received, either locally or from the network.

NOTE Unscheduled requests to transmit have no effect on the behavior of this type of AR. They are treated as “out of band” with respect to the AR behavior.

If the AREP that sent the APDU receives another APDU before the buffer contents are transmitted, the buffer contents will be replaced with the new APDU, and the previous APDU will be lost. When the buffer contents are transmitted, the AR ASE notifies the user of the transmission.

At the receiving endpoint, the APDU is received from the network and is immediately written into the buffer, completely overwriting the existing contents of the buffer. The endpoint notifies the user that the APDU has arrived and delivers it to the user according to the local user interface. If the APDU has not been delivered before the next APDU arrives, it will be overwritten by the next APDU and lost.

An FAL user receiving the buffered transmission may later request to receive the currently buffered APDU.

This type of AR also supports, as an option, the ability of the AR ASE to determine publishing, transmission, or reception timeliness. For publishing timeliness, the sending AR ASE will mark the buffer if it has not been updated within a specified update interval. For reception timeliness, the receiving AR ASE will mark the buffer if it has not been received within a specified reception interval. Both publishing and reception timeliness information are given. For transmission timeliness, the sending AR ASE determines if the buffer has been transmitted within a specified interval. If the transmission does not occur, the FAL user is notified and the buffer contents are updated to indicate the lack of transmission timeliness.

Residence timeliness is supported by this type of AREP. Figure 20 illustrates this type of timeliness.

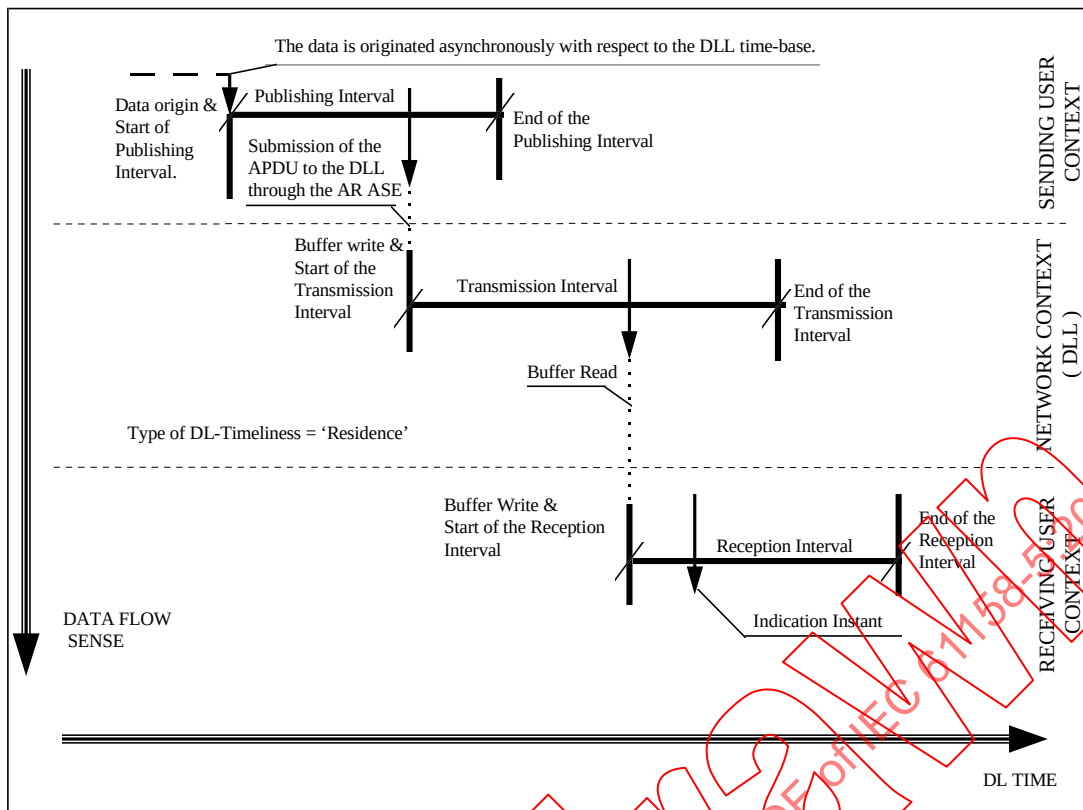


Figure 20 – Residence Timeliness

This type of timeliness requires the use of a data link timer associated with data buffer (transmit or receive). The timer is started when the data is written into the transmit buffer by the publisher AREP or received from the network into the subscriber buffer. The length of the time interval is configured for the AREP by management.

The timeliness indicator for this type of AREP is computed as follows:

Publisher AREP Data is timely if it is transmitted within the timeliness window after it is placed into the data link buffer for transmission. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent transmissions of the same data from the buffer will indicate its current status.

Subscriber AREP Data is timely if it is received as timely by the Data Link Layer and is read from the receive buffer by the AREP within the timeliness window that was started when the data was received. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent reads from the buffer of the same data will indicate its current status.

Synchronized timeliness is supported by this type of AREP. Figure 21 illustrates this type of timeliness.

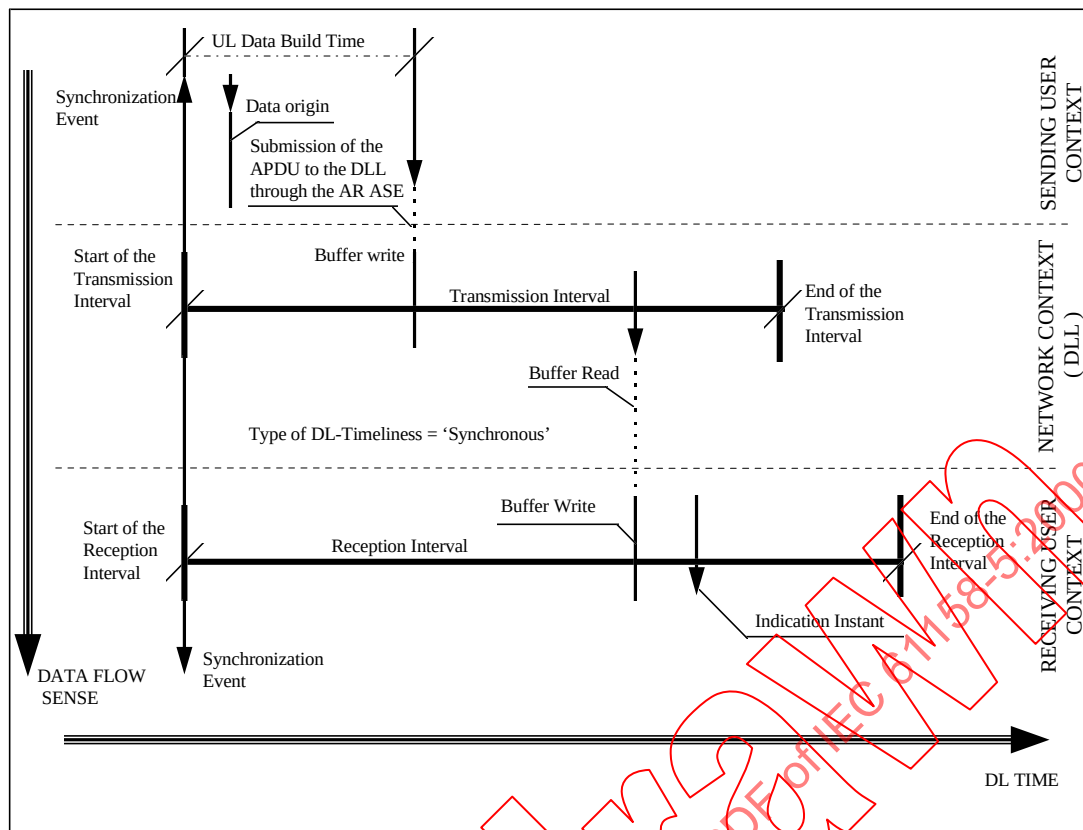


Figure 21 – Synchronized Timeliness

This type of timeliness requires the use of two supporting data link connections. One is used to publish the data and the other is used to distribute a “sync mark” within the Data Link Layer to each of the stations participating as either a publisher or a subscriber. The sync mark causes a timer associated with data buffer (transmit or receive) to be started within the Data Link Layer. The length of the time interval is configured for the AREP by management.

The timeliness indicator for this type of AREP is computed as follows:

Publisher AREP Data is timely if it is placed into the buffer within the time window after the receipt of a “sync mark”. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent transmissions of the same data from the buffer will indicate the current status of the buffer.

Subscriber AREP Data is timely if it is received as timely by the Data Link Layer and is received within the time window. When the timeliness window expires, the data in the buffer is regarded as NOT Timely. Data received after the expiration of the window, but before the next sync mark is also considered NOT Timely. Subsequent reads from the buffer of the same data will indicate its current status.

The following summarizes the characteristics of this AREP class.

Roles: Push Publisher
 Push Subscriber

Cardinality: 1-to-Many

Timeliness: Optional

7.3.7.2 Formal Class Definition

FAL ASE:	AR ASE
CLASS:	Buffered Network-Scheduled Uni-directional AR Endpoint
CLASS ID:	38
PARENT CLASS:	AR Endpoint
NETWORK MANAGEMENT ATTRIBUTES:	
1	(m) Attribute: Role (PUSH PUBLISHER, PUSH SUBSCRIBER)
2	(m) Attribute: AREP State
2	(c) Constraint: Role == PUSH SUBSCRIBER
2.1	(o) Attribute: Duplicate Pdu Detection Supported
3	(m) Attribute: DL Mapping Reference
SERVICES:	
1	(o) OpsService: Unconfirmed Send
2	(o) OpsService: AR Compel
3	(o) OpsService: Get Buffered Message
4	(o) OpsService: Schedule Communication
5	(o) OpsService: Cancel Scheduled Sequence

7.3.7.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

PUSH-PUBLISHER Endpoints of this type publish their data issuing unconfirmed service Request-APDUs.

PUSH-SUBSCRIBER Endpoints of this type receive data published in confirmed service Response-APDUs.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

DL Mapping Reference

For **PUSH-PUBLISHER** AREPs, this attribute specifies the mapping to the transmit conveyance path. For **PUSH-SUBSCRIBER** AREPs, this attribute specifies the mapping to the receive conveyance path. DL mapping attributes for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

7.3.7.2.2 Services

Unconfirmed Send

This optional service is used to send an unconfirmed service on an AR.

AR Compel

This optional service is used to cause an APDU locally buffered for transmission (for **PUSH-PUBLISHER** AREPs) or remotely buffered for transmission (for **PUSH-SUBSCRIBER** AREPs) in the Data Link Layer to be transmitted at the next available opportunity.

Get Buffered Message

This local service is used to retrieve an APDU from the buffer used by an AR.

Schedule Communication

This local service provides the AL user with the ability to schedule a sequence of DL-COMPELS for the AREP.

Cancel Scheduled Sequence

This local service provides the AL user with the ability to cancel an existing sequence which has previously been scheduled for the AREP.

7.3.8 Buffered Network-scheduled Bi-directional (BNB) AR Endpoint Formal Model

7.3.8.1 Description

This class is defined to support the cyclic exchange of confirmed services (only Read) and on-demand exchange of unconfirmed services between two application processes using buffers. The cycle is given by the network.

The behavior of this type of AR can be described as follows.

An AR ASE user wishing to convey a Read request or unconfirmed request APDU submits it as an AR ASE Service Data Unit to its AREP. The AREP sending the request APDU writes it into the Data Link Layer buffer, completely replacing the existing contents of the buffer. Furthermore, a copy of the APDU is stored at a local buffer within the AREP if the Data Link Layer buffer is currently used by a precede request. The Data Link Layer transfers the buffer contents once at the earliest transfer by opportunity when a request to transmit (AR Compel) is received from the network.

At the receiving endpoint, the APDU is received from the network, it is written into the buffer, completely replacing the existing contents of the buffer. The endpoint notifies the user that the confirmed APDU (Read) has arrived and delivers it to the user according to the local user interface. Furthermore, a copy of the APDU is stored at a local buffer within the AREP. The following response from the AR ASE user is written by the AREP into the Data Link Layer buffer, completely replacing the existing contents of the buffer. The Data Link Layer transfers the buffer contents once at the earliest transfer opportunity when a request to transmit (AR Compel) is received from the network. The AR ASE user is notified again with the restored request when the response is sent over the network. This causes the AR ASE user to respond again the current requested variable in a cyclic manner.

At the requesting endpoint, the APDU containing the response received from the network, it is written to the buffer, completely replacing the existing contents of the buffer. The endpoint notifies the user that the APDU has arrived and delivers it to the user according to the local user interface as the confirmation. If the AR ASE user wishes to convey a Read request again the response is generated locally from the Data Link Layer buffer. The content of the buffer is cyclic updated by the network with the related mechanism at the receiving endpoint.

The AREP receiving an unconfirmed request APDU from its underlying layer, queues it for delivery to its AR ASE user in the order that it was received.

Additionally the AREP monitors the connection by observing the network activities.

The following summarizes the characteristics of this AREP class.

Roles:	Peer
Cardinality:	1-to-1
Timeliness:	No

7.3.8.2 Formal Class Definition

FAL ASE: **AR ASE**
CLASS: Buffered Network-Triggered Bi-directional AR Endpoint
CLASS ID: 44
PARENT CLASS: AR Endpoint

NETWORK MANAGEMENT ATTRIBUTES:

1	(m)	Attribute:	Role (CLIENT, SERVER, PEER)
2	(m)	Attribute:	AREP State
3	(m)	Attribute:	Remote Address Configuration Type (FREE, LINKED)
4	(m)	Attribute:	Initiator (TRUE/FALSE)
5	(m)	Attribute:	Receive DL Mapping Reference
6	(m)	Attribute:	Transmit DL Mapping Reference
7	(o)	Attribute:	Remote AP Name
8	(m)	Attribute:	CIN

SERVICES:

1	(o)	OpService:	Unconfirmed Send
2	(o)	OpService:	Confirmed Send
3	(o)	OpService:	Establish
4	(o)	OpService:	Abort

7.3.8.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

CLIENT, SERVER, PEER () Endpoints of this type may perform as either clients or servers, or both simultaneously. Endpoints of this type must indicate whether or not they are to be the initiator of the AR establishment process.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

Remote Address Configuration Type

This attribute specifies how the remote address of the AREP is configured. The valid values are:

FREE	The value of "FREE" indicates that the destination of the FAL-PDUs from the Source AREP is provided dynamically in the request service.
LINKED	The value of "LINKED" indicates that the destination of the FAL-PDUs is configured with the RemoteDlsapAddress attribute contained in the DL Mapping (specified in IEC 61158-6).

Initiator

This attribute indicates, when TRUE, that the endpoint has been configured to initiate the establishment of the AR. One and only one of the AREPs involved in an AR must be the initiator. When this AREP is a client, the value of this attribute is always TRUE. When this AREP is a server, the value of this attribute is always FALSE. When this AREP is a Peer, the value of this attribute may be either, as long as both AREPs in the AR are not configured to be the initiator.

Receive DL Mapping Reference

This attribute provides a reference to the underlying Data Link Layer mapping for the conveyance path from the Publishing Manager. DL mapping attributes for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

Transmit DL Mapping Reference

This conditional attribute provides a reference to the underlying Data Link Layer mapping for the transmit conveyance path. It is present only for PULL PUBLISHERs and PULL PUBLISHING MANAGERS. DL mapping attributes for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

Remote AP Name

This optional attribute specifies the name of the AP attached at the remote AREP.

CIN

This attribute (CIN - Control Interval Network triggered) indicates the control interval for monitoring a network triggered connection.

7.3.8.2.2 Services**Unconfirmed Send**

This optional service is used to send an unconfirmed service on an AR.

Confirmed Send

This optional service is used to send a confirmed service on an AR.

Establish

This service is used to establish an AR.

DeEstablish

This service is used to de-establish (gracefully terminate) an AR.

Abort

This service is used to terminate an AR.

7.3.9 Buffered Network-Scheduled And Unscheduled Uni-directional (BNU-Mp) AR Endpoint Formal Model

7.4 Description

This class is defined to support the “push” model for scheduled, buffered distribution of unconfirmed services to one or more application processes.

The behavior of this type of AR can be described as follows.

The push model is used for the scheduled traffic, as follows.

An AR ASE user wishing to broadcast new data, uses an AR ASE Service Data Unit with a special encoding (see IEC 61158-6) to its AREP for distribution. The AREP sending the data writes it into the Data Link Layer buffer, completely replacing the existing contents of the buffer. The Data Link Layer transfers the buffer contents at the next scheduled opportunity. When the buffer contents are transmitted, the AR ASE notifies the user of the transmission.

If the AREP that sent data receives from the AR ASE user another data before the buffer contents are transmitted, the buffer contents will be replaced with the new data and the previous data will be lost.

At the receiving endpoint, the data is received from the network, it is immediately written into the buffer, completely overwriting the existing contents of the buffer. The endpoint notifies the user that the data has arrived and delivers it to the user. Extra local readings of the contents of the buffer are possible. If the data has not been delivered before the next data arrives, it will be overwritten by the next data and lost.

The pull model is used for the on-demand traffic, as follows.

An AR ASE user wishing to send or read the remote value of the buffer, uses the Remote-Write or Remote-Read services with an AR ASE Service Data Unit specially encoded (see IEC 61158-6). An AR ASE user wishing to broadcast data on-demand writes it into the Data Link Layer buffer replacing the existing contents of the buffer. The Data Link Layer transfers the buffer contents after soliciting a transmission opportunity. The transmission of the buffer is reported to the user as the confirmation of the service. At the receiving endpoint, the data is received from the network and is immediately written into the buffer, completely overwriting the existing contents of the buffer. The endpoint notifies the user that the data has arrived and delivers it to the user. Extra local readings of the contents of the buffer are possible.

An AR ASE user wishing to read the remote value of a buffer, asks the Data Link Layer to solicit a transmission opportunity to get the value of the remote buffer. The transmission of the buffer is reported to the user as the confirmation of the service. At the receiving endpoint, the data is received from the network and is immediately written into the buffer, completely overwriting the existing contents of the buffer. The endpoint notifies the user that the data has arrived and delivers it to the user. Extra local readings of the contents of the buffer are possible.

This type of AR also supports, as an option, the ability of the AR ASE to determine publishing, transmission, or reception timeliness. For publishing timeliness, the sending AR ASE will mark the buffer if it has not been updated within a specified update interval. For reception timeliness, the receiving AR ASE will mark the buffer if it has not been received within a specified reception interval. Both publishing and reception timeliness information are given. For transmission timeliness, the sending AR ASE determines if the buffer has been transmitted within a specified interval. If the transmission does not occur, the FAL user is notified and the buffer contents are updated to indicate the lack of transmission timeliness.

Residence timeliness is supported by this type of AREP. Figure 22 illustrates this type of timeliness.

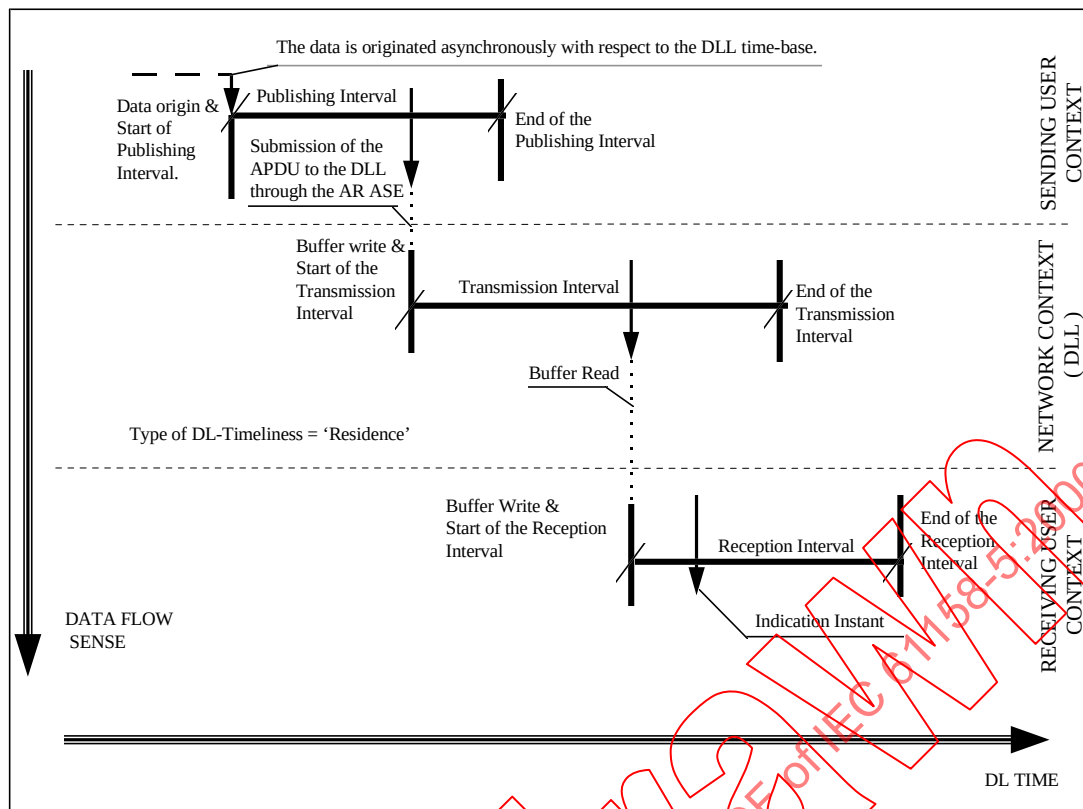


Figure 22 – Residence Timeliness

This type of timeliness requires the use of a data link timer associated with data buffer (transmit or receive). The timer is started when the data is written into the transmit buffer by the publisher AREP or received from the network into the subscriber buffer. The length of the time interval is configured for the AREP by management.

The timeliness indicator for this type of AREP is computed as follows.

Publisher AREP Data is timely if it is transmitted within the timeliness window after it is placed into the data link buffer for transmission. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent transmissions of the same data from the buffer will indicate its current status.

Subscriber AREP Data is timely if it is received as timely by the Data Link Layer and is read from the receive buffer by the AREP within the timeliness window that was started when the data was received. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent reads from the buffer of the same data will indicate its current status.

Synchronized timeliness is supported by this type of AREP. Figure 23 illustrates this type of timeliness.

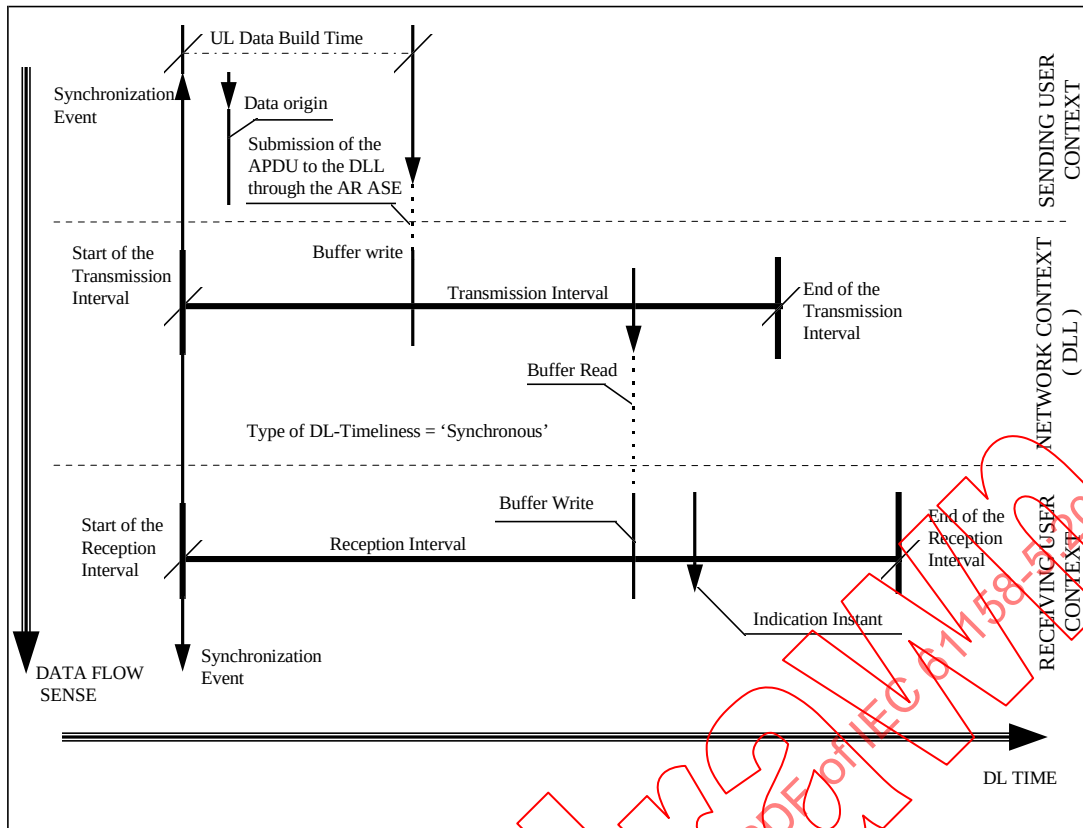


Figure 23 – Synchronized Timeliness

This type of timeliness requires the use of two supporting data link connections. One is used to publish the data and the other is used to distribute a “sync mark” within the Data Link Layer to each of the stations participating as either a publisher or subscriber. The sync mark causes a timer associated with data buffer (transmit or receive) to be started within the Data Link Layer. The length of the time interval is configured for the AREP by management.

The timeliness indicator for this type of AREP is computed as follows.

Publisher AREP data is timely if it is placed into the buffer within the time window after the receipt of a “sync mark”. When the timeliness window expires, the data in the buffer is regarded as NOT timely. Subsequent transmissions of the same data from the buffer will indicate the current status of the buffer.

Subscriber AREP Data is timely if it is received as timely by the Data Link Layer and is received within the time window. When the timeliness window expires, the data in the buffer is regarded as NOT Timely. Data received after the expiration of the window, but before the next sync mark, is also considered NOT Timely. Subsequent reads from the buffer of the same data will indicate its current status.

The following summarizes the characteristics of this AREP class.

Roles: Push Publisher
Push Subscriber

Cardinality: 1-to-Many

Timeliness: Optional

L.1 Formal Class Definition

FAL ASE:

AR ASE

CLASS:

Buffered Network-Scheduled Uni-directional AR Endpoint

CLASS ID:

39

PARENT CLASS:

AR Endpoint

NETWORK MANAGEMENT ATTRIBUTES:

1	(m)	Attribute:	Role	(PUSH PUBLISHER, PUSH SUBSCRIBER, PULL PUBLISHER, PULL SUBSCRIBER)
2	(m)	Attribute:	AREP State	
3	(m)	Attribute:	DL Mapping Reference	

SERVICES:

1	(o)	OpService:	Unconfirmed Send
2	(o)	OpService:	AR Compel
3	(o)	OpService:	Get Buffered Message
4	(o)	OpService:	Schedule Communication
5	(o)	OpService:	Cancel Scheduled Sequence
6	(o)	OpService:	Remote Read
7	(o)	OpService:	Remote Write

7.4.1.1.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

PUSH-PUBLISHER Endpoints of this type publish their data issuing unconfirmed service Request-APDUs.

PUSH-SUBSCRIBER Endpoints of this type receive data published in confirmed service Response-APDUs.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

DL Mapping Reference

For PUSH-PUBLISHER AREPs, this attribute specifies the mapping to the transmit conveyance path. For PUSH-SUBSCRIBER AREPs, this attribute specifies the mapping to the receive conveyance path. DL mapping attributes for the Data Link Layer (IEC 61158-3) are specified in IEC 61158-6.

7.4.1.1.2 Services

Unconfirmed Send

This optional service is used to send an unconfirmed service on an AR. The structure of the PDU is specified and described in the corresponding DL-Mapping State Machine.

AR Compel

This optional service is used to cause an APDU locally buffered for transmission (for PUSH-PUBLISHER AREPs) or remotely buffered for transmission (for PUSH-SUBSCRIBER AREPs) in the Data Link Layer to be transmitted at the next available opportunity.

Get Buffered Message

This local service is used to retrieve an APDU from the buffer used by an AR.

Schedule Communication

This local service provides the AL user with the ability to schedule a sequence of DL-COMPELS for the AREP.

Cancel Scheduled Sequence

This local service provides the AL user with the ability to cancel an existing sequence which has previously been scheduled for the AREP.

Remote Read

This service (based on the Pull Subscriber model) allows a user to read the contents of a remote buffer. The structure of the PDU is described and specified in the corresponding DL-Mapping State Machine in the protocol document.

Remote Write

This service (based on the Pull Subscriber model) allows a user to write the contents of a remote buffer. The structure of the PDU is specified and described in the corresponding DL-Mapping State Machine in the protocol document.

7.5 Summary of Type 1 FAL Classes

This clause contains a summary of the defined FAL Classes. The Class ID values have been assigned to be compatible with existing standards. Table 72 provides a summary of the Type 1 classes.

Table 72 – Type 1 FAL Class Summary

FAL ASE	Class	Class ID
Application Process	Application Process	16
Data Type	Fixed Length & String Data Type	5
	Structure Data Type	6
	Array Data Type	12
Application Relationship	AREP	32
	QUB-Co	34
	QUU	36
	BNU	38
	BNU Mp	39
	BUU (for future study)	40
	BUB	42
	BNB	44
	QUB-CI	45
	QUB-FC	46
	QUB-Seg	45
Variable	Fixed Length & String Variable	7
	Array Variable	8
	Data Structure Variable	9
	Variable List	10
Event	Event	4
	Event List	17
	Notifier	18
Load Region	Load Region	2
Function Invocation	Function Invocation	3
	Action	19

7.6 Permitted FAL Type 1 Services by AREP Role

Table 73 below defines the valid combinations of services and AREP roles (which service APDUs and AREP with the specified role can send or receive). The Unc and Cnf columns indicate whether the service listed in the left-hand column is unconfirmed (Unc) or confirmed (Cnf).

Table 73– Services by AREP Role

	Unc	Cnf	Client	Server	Push Publ	Push Subsc	Pull Publ Mgr	Pull Publ	Pull Subsc	Report Src	Report Sink
FAL Services			req rcv	req rcv	req rcv	req rcv	req rcv	req rcv	req rcv	req rcv	req rcv
Mgt ASE											
Create		X	X	X							
Delete		X	X	X							
Get Attributes		X	X	X							
Get Attribute List		X	X	X							
Set Attributes		X	X	X							
Begin Set Attributes		X	X	X							
End Set Attributes		X	X	X							
AP ASE											
Subscribe		X	X	X							
Identify		X	X	X							
Get Status		X	X	X							
Status Notification	X				X	X				X	X
Initiate		X	X	X							
Terminate		X	X	X							
Conclude		X	X	X							
Reject		X	X								
AR ASE											
AR-Confirmed Send			X	X							
AR-Unconfirmed Send			X X	X X	X	X					
AR-Establish			X	X							
AR-De-Establish			X	X							
AR-Abort			X X	X X	X	X X				X	X
AR Compel			X	X		X					
AR-Get Buffered Msg			X	X		X					
AR-Schedule					X	X					
Communication					X	X					
Sequence											
AR-Status			X	X							
AR-XON-OFF			X	X							
AR Remote Read							X	X	X		
AR Remote Write							X	X	X		
Variable ASE											
Read		X	X	X			X X	X			
Read List		X	X	X							
Write		X	X	X							
Write List		X	X	X							
Information Report	X				X	X				X	X
Information Report List	X				X	X				X	X
Exchange		X	X	X							
ExchangeList		X	X	X							

(continued on next page)

8 Communication Model Type 2 Specification

8.1 Type 2 Concepts

8.1.1 General

Type 2 is a serial communication system for communication between devices that wish to exchange time critical application information in a deterministic and predictable manner. These devices include simple I/O devices, such as sensors/actuators as well as complex control devices such as robots, programmable logic controllers, welders, process controllers, etc.

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

Type 2 provides a high degree of protocol efficiency by utilising an implied token passing mechanism. This mechanism allows all devices equal access to the network without the network overhead associated with passing a "token" to each device granting it permission to send data. The protocol utilises a time based scheduling mechanism which provides network devices with deterministic and predictable access to the medium while preventing network collisions. This scheduling mechanism allows time critical data, which is required on a periodic, repeatable and predictable basis, to be produced on a predefined schedule without the loss of efficiency associated with continuously requesting or "polling" for the required data.

The network protocol supports an additional mechanism which allows data that is not time critical in nature or which is only required on an occasional basis to utilise any available network time. This unscheduled data is transmitted after the production of the time critical data has been completed and before the beginning of the next scheduled production of time critical data.

8.1.2 General concepts

Type 2 uses most of the general concepts described in clause 4, with some differences as noted below :

- AR follow a model which is detailed in 8.3.1.
- Client/Server relationships can be one-to-one, but also one-to-many (see AR model in 8.3.1)
- An additional Producer/Consumer model is used as the base of all AR's (see AR model in 8.3.1)
- Type 2 does not provide any kind of Abort or Reject services, only negative result confirmations, so the error procedure described in Annex A does not apply.
- Specific ranges and values are defined for Class ID's codes and error codes for Type 2 (in IEC 61158-6)
- Specific naming and addressing is specified in 8.1.4.

8.1.3 Relationships between ASE's

Every node in the Type 2 shall contain as a minimum instance 1 of the following ASE object classes in its application layer :

Object Management ASE :

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

Connection Manager ASE :

- Connection Manager (establishment and maintenance of connections),

AR ASE :

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

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

Object Management ASE :

- Assembly Object (binds data from multiple objects to be sent through one connection)

Yet more Type 2 objects belong to the Type 2 system management, primarily concerned with the Data Link layer :

- ControlNet Object (station management interfaces to lower layers, required in all devices),
- Keeper Object (holds and distributes attributes of all devices on the link, one required per link, with optional backup(s)),
- Scheduling Object (holds information on link schedules, one required in each connection originator).

NOTE These three Management objects will be described in the future IEC 61158-7.

The ASE and object interactions are illustrated in Figure 24:

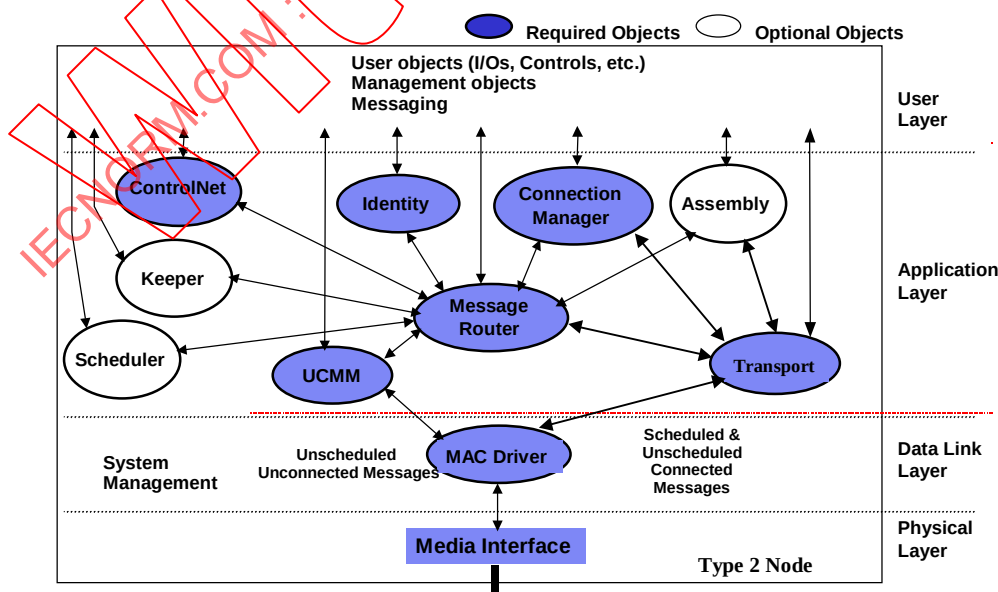


Figure 24 – Overview of ASE's and object classes

8.1.4 Naming and Addressing

The information presented here provides a common basis for logically addressing separate physical components across the network for Type 2. These addressing terms are used in the Type 2 specifications. Figure 25 should be referenced in the following discussion.

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

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

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

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

The address of attribute number 1 of instance 2 in class 5 in the node with a MAC ID of 4 is MAC ID#4: Object Class #5: Instance #2: Attribute #1 as shown in Figure 25. This nomenclature is referred to as Class/Instance/Attribute addressing.

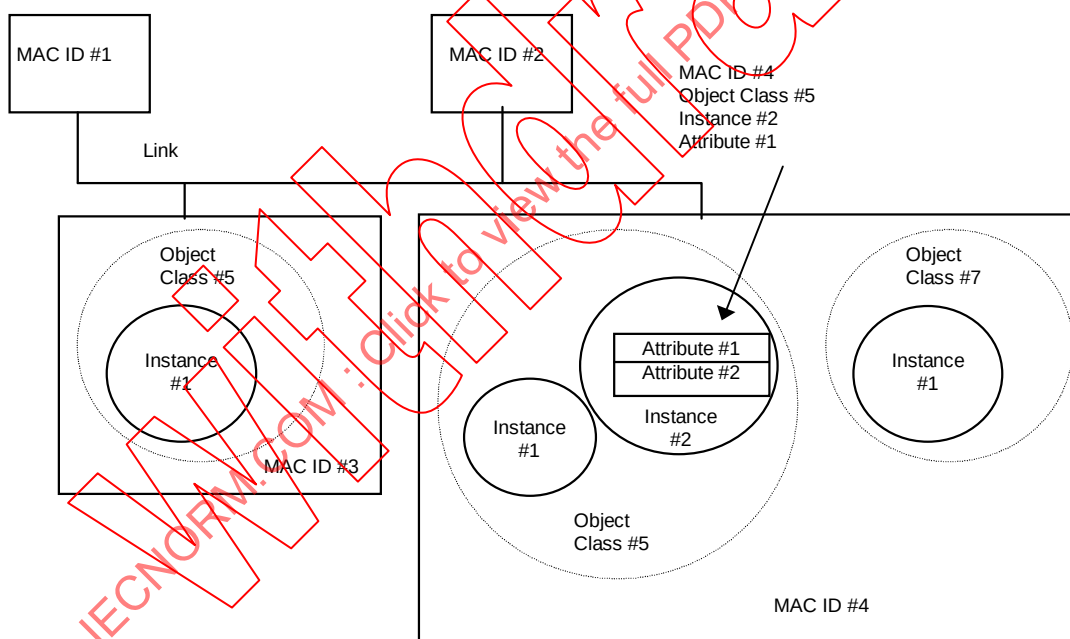


Figure 25 – Addressing format using MAC, Class, Instance and Attribute IDs

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

8.1.5 Type 2 Data Types

8.1.5.1 Type 2 Data Type General Specification

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

Data used together with Type 2 shall be made up of elementary (primitive) data types. These elementary data types shall be used to construct derived (constructed) data types.

NOTE1: Type 2 specifies elementary and derived data types corresponding to the notation of IEC 61131-3. In addition, since function blocks as defined in IEC 61131-3 have both an associated data structure and a set of defined operations, these elements are also specified below as additional data types for Type 2.

The derived data types for Type 2 shall be :

- directly derived;
- enumerated;
- subrange;
- structured;
- array.

NOTE2: These derived data types are defined in 1.3 and 2.3.3 of IEC 61131-3. The means of specifying these data types and their default initial values is defined in 2.3.3.1 and 2.3.3.2 of IEC 61131-3. The usage of variables of these data types is defined in 2.3.3.3 of IEC 61131-3.

8.1.5.2 Supported Elementary Data Types

Type 2 supports the following basic data types defined in clause 5.

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

8.1.5.3 Compliance with IEC 61131-3

8.1.5.3.1 Compliance statement

NOTE1: This clause provides information with respect to only the data types and associated operations defined in IEC 61158-5 for Type 2.

NOTE2: Subclause 1.5.1 of IEC 61131-3 defines the requirements which are met by programmable controller systems claiming compliance to IEC 61131-3. This provides the data type-related information to be included in the documentation of functional units which support the data types defined in IEC 61158-5. Subsets or extensions of this documentation are provided as appropriate to the specific compliant functional unit.

Type 2 shall comply with the requirements of IEC 61131-3 for language features as specified in Table 74, Table 75, and Table 76.

Table 74 – Common elements

IEC 61131-3 Table/feature	Feature description
10/1	BOOL data type
10/2	SINT data type
10/3	INT data type
10/4	DINT data type
10/5	LINT data type
10/6	USINT data type
10/7	UINT data type
10/8	UDINT data type
10/9	ULINT data type
10/10	REAL data type
10/11	LREAL data type
10/12	TIME data type
10/13	DATE data type
10/14	TIME_OF_DAY or TOD data type
10/15	DATE_AND_TIME or DT data type
10/16	STRING data type
10/17	BYTE data type
10/18	WORD data type
10/19	DWORD data type
10/20	LWORD data type
12/1	Directly derived data types
12/2	Enumerated data types
12/3	Subrange data types
12/4	Array data types
12/5	Structured data types
13	Standard default initial values
Subclause 2.5.1.3	User declared functions (no table entry)
22/1	Type conversions (see Table 76)
22/2	TRUNC function
22/3	BCD_TO_** functions
22/4	*_TO_BCD functions
23/111	Standard functions of one numeric variable: ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN
24/ 12n18n	Standard named arithmetic functions: ADD, MUL, SUB, DIV, MOD, EXPT, MOVE
24/ 12s15s, 17s,18s	Standard symbolic arithmetic functions: +, *, -, /, **, :=
25/14	Standard bit string functions: SHL, SHR, ROR, ROL
26/ 5s8s	Standard named bitwise Boolean functions: AND, OR, XOR, NOT
26/ 5s7s	Standard symbolic bitwise Boolean functions: &, >=1, =2k+1
27/14	Standard selection functions: SEL, MAX, MIN, LIMIT, MUX
28/5n10n	Standard named comparison functions: GT, GE, EQ, LE, LT, NE
28/5s10s	Standard symbolic comparison functions: >, >=, =, <=, <, <>

(continued on next page)

29/ 19	Standard character string functions: LEN, LEFT, RIGHT, MID, CONCAT, INSERT, DELETE, REPLACE, FIND
30/ 114	Standard functions of time data types: ADD, SUB, MUL, DIV, CONCAT, DATE_AND_TIME_TO_TIME_OF_DAY, TIME_OF_DAY_TO_DATE_AND_TIME NOTE Table 30 of IEC 61131-3 limits the data types to which these operations apply.
31/14	Standard functions of enumerated data types: SEL, MUX, EQ, NE
32	Standard access mechanisms to function block I/O parameters
33/8a,8b,9a,9b	Userdefined function blocks per 2.5.2.2 of IEC 61131-3, with graphical or textual rising or falling edge input options
34/13	Standard bistable function blocks: SR, RS, SEMA
35/1,2	Standard edge detection function blocks: R_EDGE, F_EDGE
36/13	Standard counter function blocks: CTU, CTD, CTUD
37/1,2a,3a, 4	Standard timer function blocks: TP, TON, TOF, RTC
55/ 117	Standard operators: (), function evaluation, **, -, NOT, *, /, MOD, +, -, <, >, <=, >=, =, <>, &, AND, XOR, OR

Table 75 – ST language elements

IEC 61131-3 Table/feature	Feature description
55/117	Standard operators: (), function evaluation, **, -, NOT, *, /, MOD, +, -, <, >, <=, >=, =, <>, &, AND, XOR, OR
56/2	Function block invocation and output usage

NOTE to Table 75:

For detail information, see the notes following Table 76.

Table 76 – Type conversion operations

Operation	Result	Error conditions
ANY_BIT_TO_ANY_BIT	See note 4	None
ANY_BIT_TO_ANY_INT	OUTmin + Sbk2k See note 5	Result > OUTmax
ANY_BIT_TO_BOOL	FALSE if IN = 0 TRUE otherwise	None
ANY_BIT_TO_STRING	See note 6	None
ANY_DATE_TO_STRING	See note 7	None
ANY_INT_TO_BOOL	FALSE if IN = 0 TRUE otherwise	None
ANY_NUM_TO_ANY_INT	IN See note 8	(IN > OUTmax) or (IN < OUTmin)
ANY_NUM_TO_ANY_REAL	IN	See note 9
ANY_NUM_TO_DATE	See note 10	
ANY_NUM_TO_STRING	See note 11	
ANY_NUM_TO_TIME	See note 12	
ANY_NUM_TO_TOD	See note 13	
ANY_REAL_TO_BOOL	FALSE if IN = 0.0 TRUE otherwise	None
BOOL_TO_ANY_BIT BOOL_TO_ANY_INT	0 if IN = FALSE 1 if IN = TRUE	None
BOOL_TO_ANY_REAL	0.0 if IN = FALSE 1.0 if IN = TRUE	None
BOOL_TO_STRING	'FALSE' if IN = FALSE 'TRUE' if IN = TRUE	None
DATE_TO_ANY_NUM	See note 14	Result > OUTmax
STRING_TO_ANY	Converted data	See note 15
TIME_TO_ANY_NUM	See note 16	Result > OUTmax
TOD_TO_ANY_NUM	See note 17	Result > OUTmax

NOTES to Table 75 and Table 76:

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

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

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

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

```
BYTE_TO_WORD(16#FF) = 16#00FF
and
WORD_TO_BYTE(16#0FF0) = 16#F0
```

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

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

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

NOTE 8 Conversion of REAL and LREAL to integer types is accomplished by rounding as specified in 5.4 of IEEE 754.

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

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

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

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

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

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

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

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

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

8.1.5.3.2 Implementation dependant parameters

The values of implementation dependent parameters from Table D.1 of IEC 61131-3 that are defined for Type 2 shall be as shown in Table 77.

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

Table 77 – Values of implementation-dependent parameters

Subclause of IEC 61131-3	Parameter	Value
2.2.3.1	Range of values of duration	Same as LINT in microseconds
2.3.1	Range of values for variables of type TIME	Same as DINT in milliseconds
	Precision of representation of seconds in types TIME_OF_DAY and DATE_AND_TIME	1 millisecond
2.3.3.1	Maximum number of enumerated values	256
2.3.3.2	Default maximum length of STRING variables	256
	Maximum allowed length of STRING variables	65536
2.4.1.2	Maximum number of subscripts	8
	Maximum range of subscript values	0 – 255
	Maximum number of levels of structures	8
2.5.1.5	Maximum inputs of extensible functions	8
2.5.1.5.1	Effects of type conversions on accuracy	As defined in Table 76.
2.5.1.5.2	Accuracy of functions of one variable	As defined in IEEE 754
2.5.2.3.3	PVmin, PVmax of counters	0, 65535

8.1.5.3.3 Language extensions

The extensions to IEC 61131-3 defined for Type 2 shall be as listed in Table 78. When these extensions are used in a particular device, the clause references in this table shall be replaced by references to the descriptions of the corresponding extensions in the functional unit's documentation.

Table 78 – Extensions to IEC 61131-3

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

8.2 Type 2 ASEs

8.2.1 Type 2 Object Management ASE

8.2.1.1 Overview

The FAL Type 2 Object Management ASE is used to access all network accessible application elements or system management elements.

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

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

8.2.1.2 FAL Management Model Class Specification

8.2.1.2.1 General Type 2 Formal Model

The base AP class below specifies the common attributes and services defined for all type 2 AP classes, including those defined by the user (application specific).

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

FAL ASE: **FAL Management ASE**

CLASS: Base_Object

CLASS ID: <tbid>

PARENT CLASS: TOP

ACCESS ATTRIBUTES:

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

SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES) :

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

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES) :

- 1 (o) Attribute: Attr1

SYSTEM MANAGEMENT SERVICES:

- 1 (o) Mgt Service: Get_Attribute_All
- 2 (o) Mgt Service: Set_Attribute_All
- 3 (o) Mgt Service: Get_Attribute_List
- 4 (o) Mgt Service: Set_Attribute_List
- 5 (o) Mgt Service: Reset
- 6 (o) Mgt Service: Start
- 7 (o) Mgt Service: Stop
- 8 (o) Mgt Service: Create
- 9 (o) Mgt Service: Delete
- 10 (o) Mgt Service: Get_Attribute_Single
- 11 (o) Mgt Service: Set_Attribute_Single
- 12 (o) Mgt Service: Find_Next_Object_Instance

OBJECT MANAGEMENT SERVICES:

- 1 (o) Ops Service: Get_Attribute_All
- 2 (o) Ops Service: Set_Attribute_All
- 3 (o) Ops Service: Get_Attribute_List
- 4 (o) Ops Service: Set_Attribute_List
- 5 (o) Ops Service: Reset
- 6 (o) Ops Service: Start
- 7 (o) Ops Service: Stop
- 8 (o) Ops Service: Create
- 9 (o) Ops Service: Delete
- 10 (o) Ops Service: Get_Attribute_Single
- 11 (o) Ops Service: Set_Attribute_Single

OBJECT SPECIFIC SERVICES:

- 1 (o) OpsService: Service1

8.2.1.2.1.1 Access Attributes

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

Object Instance number

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

Symbolic name

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

8.2.1.2.1.2 System Management Attributes (Class Attributes)

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

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

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

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

Revision

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

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

Max Instance

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

Number of Instances

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

Optional attribute list

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

Number of attributes

Number of attributes in the optional attribute list.

Optional attributes

List of optional attribute numbers.

Optional service list

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

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

Number services

Number of services in the optional service list.

Optional services

List of optional service codes

Maximum ID Number Class Attributes

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

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

Maximum ID Number Instance Attributes

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

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

8.2.1.2.1.3 Object Management Attributes

Attributes at the instance level

An Instance Attribute shall be unique to an object instance and not shared by the object class. Instance ID = 0 shall be a special case. A service directed to Instance ID = 0 shall be applied to the class, not to a particular instance.

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

8.2.1.2.1.4 System Management (Class Level) and Object Management (Instance Level) Services

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

Get_Attribute_All

The Get_Attribute_All service shall return the contents of all attributes of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

Set_Attribute_All

The Set_Attribute_All service shall modify the contents of all attributes of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

Get_Attribute_List

The Get_Attribute_List service shall return the contents of the selected gettable attributes of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

Set_Attribute_List

The Set_Attribute_List service shall update the contents of the selected attributes of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

Reset

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

Start

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

Stop

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

Create

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

Delete

The Delete service shall delete an object instance of the specified object class.

Get_Attribute_Single

The Get_Attribute_Single service shall return the contents of the specified attribute of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

Set_Attribute_Single

The Set_Attribute_Single service shall modify the contents of the specified attribute of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

Find_Next_Object_Instance

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

8.2.1.2.1.5 Object Specific Services

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

Object-specific services sent to the Class Level of an object shall be called Object-specific System Management (Class Level) Services. Object-specific services sent to the Instance Level of an object shall be called Object-specific Object Management (Instance Level) Services.

8.2.1.2.2 Identity Formal Model

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

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

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

FAL ASE: **FAL Management ASE**
CLASS: Identity_Object
CLASS ID: <td>
PARENT CLASS: Base_Object

ACCESS ATTRIBUTES:

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

SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES) :

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

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES) :

- 1 (m) Attribute: Manufacturer ID
- 2 (m) Attribute: Device Type
- 3 (m) Attribute: Product Code
- 4 (m) Attribute: Revision
- 4.1 (m) Attribute: Major Revision
- 4.2 (m) Attribute: Minor Revision
- 5 (m) Attribute: Status
- 5.1 (m) Attribute: Owned
- 5.2 (m) Attribute: Configured
- 5.3 (m) Attribute: Additional Status
- 5.4 (m) Attribute: Minor Recoverable Fault
- 5.5 (m) Attribute: Minor Unrecoverable Fault
- 5.6 (m) Attribute: Major Recoverable Fault
- 5.7 (m) Attribute: Major Unrecoverable Fault
- 6 (m) Attribute: Serial Number
- 7 (m) Attribute: Product Name

SYSTEM MANAGEMENT SERVICES:

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

OBJECT MANAGEMENT SERVICES:

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

8.2.1.2.2.1 System Management Attributes (Class Attributes)

Max Instance

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

8.2.1.2.2.2 Object Management Attributes

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

1. Manufacturer;
2. Device Type;
3. Product Code;
4. Revision.

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

Manufacturer ID

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

This instance attribute has an access rule of Get only.

Device Type

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

This instance attribute has an access rule of Get only.

Product Code

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

This instance attribute has an access rule of Get only.

Revision

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

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

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

This instance attribute has an access rule of Get only.

Major Revision

The major revision shall be incremented by the manufacturer when there is a significant change to the 'fit, form, or function' of the product. Any changes that effect the configuration choices available to the user require incrementing the major revision.

Minor Revision

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

Status

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

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

This instance attribute has an access rule of Get only.

Owned

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

Configured

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

Additional Status

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

- Self-Testing (power-up) or State Unknown
- Firmware Update in progress
- Communication Fault (lost connection)
- Unkeyed, Awaiting Connection
- Non-Volatile Configuration Bad
- Major Fault
- Connected, Active
- Idle (program mode type)

Minor Recoverable Fault

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

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

Minor Unrecoverable Fault

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

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

Major Recoverable Fault

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

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

Major Unrecoverable Fault

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

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

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

Serial Number

Serial number of device. This attribute provides a number used in conjunction with the Manufacturer ID to form a unique identifier for each device. Each manufacturer shall be responsible for guaranteeing the uniqueness of the serial number across all of its devices.

This instance attribute has an access rule of Get only.

Product Name

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

This instance attribute has an access rule of Get only.

8.2.1.2.2.3 System Management (Class Level) and Object Management (Instance Level) Services

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

Reset

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

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

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

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

8.2.1.2.3 Assembly Formal Model

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

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

Assembly ASEs (objects) shall be either dynamic or static.

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

NOTE : Static assemblies can usually be implemented entirely in ROM.

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

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

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

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

FAL ASE: **FAL Management ASE**
CLASS: Assembly_Object
CLASS ID: <tbid>
PARENT CLASS: Base_Object

ACCESS ATTRIBUTES:

- 1 (m) Key Attribute: Object Instance number
- 2 (o) Key Attribute: Symbolic name
- 3 (m) Attribute: Assembly type (STATIC, DYNAMIC)

SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES) :

- 1 (m) Attribute: Revision = 2
- 2 (o) Attribute: Max Instance

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES) :

- 1 (*) Attribute: Number of Members in List
- 2 (*) Attribute: Member List
- 2.1 (m) Attribute: Member Data Size
- 2.2 (m) Attribute: Member Path Size
- 2.3 (m) Attribute: Member Path
- 3 (m) Attribute: Data
- 4 (o) Attribute: Size

SYSTEM MANAGEMENT SERVICES:

- 8 (c) Constraint: Assembly type = DYNAMIC
- 8.1 (m) Mgt Service: Create
- 9 (c) Constraint: Assembly type = DYNAMIC
- 9.1 (o) Mgt Service: Delete
- 10 (m) Mgt Service: Get_Attribute_Single

OBJECT MANAGEMENT SERVICES:

- 9 (c) Constraint: Assembly type = DYNAMIC
- 9.1 (m) Ops Service: Delete
- 10 (m) Ops Service: Get_Attribute_Single
- 11 (*) Ops Service: Set_Attribute_Single

8.2.1.2.3.1 System Management Attributes (Class Attributes)

No specific requirement for this object.

8.2.1.2.3.2 Object Management Attributes

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

Number of Members in List

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

Number of members in "Member List" attribute below.

This instance attribute has an access rule of Get only.

Member List

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

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

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

Member Data Size

Size of member data (in bits).

Member Path Size

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

Member Path

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

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

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

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

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

No checking shall be done by the Assembly Object at any time to verify that the size of the member data is correct for the given member path. It shall be the responsibility of the assembly member to properly handle too much or too little data. The assembly shall be required to deliver the configured number of bits to the member.

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

Data

All of the member data packed into one array.

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

This instance attribute has an access rule of Set.

Size

Number of bytes in "Data" attribute.

This instance attribute has an access rule of Get only.

8.2.1.2.3.3 System Management (Class Level) and Object Management (Instance Level) Services

Delete

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

Set_Attribute_Single

The Set_Attribute_Single service shall modify the contents of the specified attribute of the specified Assembly object instance. This service is **optional** for a Static Assembly, **mandatory** for a Dynamic Assembly. This service may be used to add or remove a member to/from the Assembly Member List of a Dynamic Assembly.

8.2.1.2.3.4 Assembly State Machines

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

Static Assembly State Machine

Figure 26 and Table 79 illustrate the behaviour of Static Assembly objects.

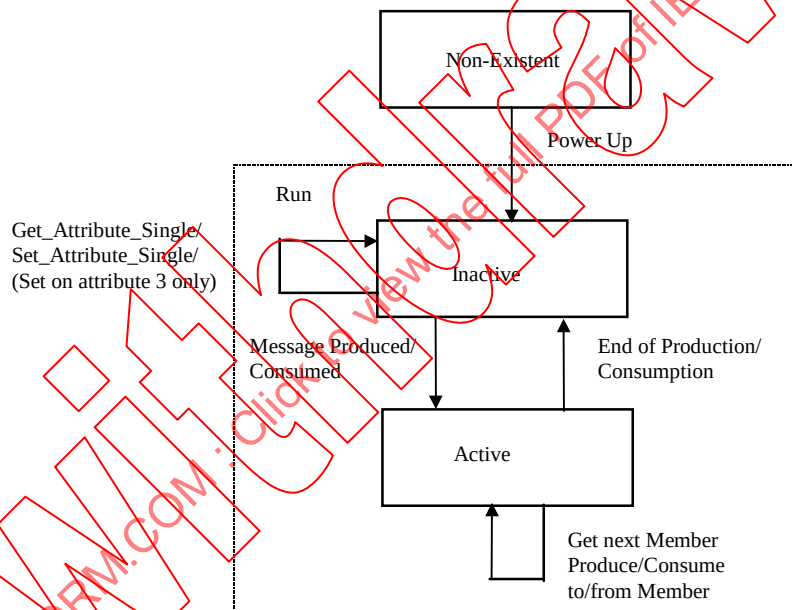


Figure 26 – Static Assembly State Transition Diagram

Table 79 – Static Assembly State Event Matrix

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

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

Dynamic Assembly State Machine

Figure 27 and Table 80 illustrate the behaviour of Dynamic Assembly objects.

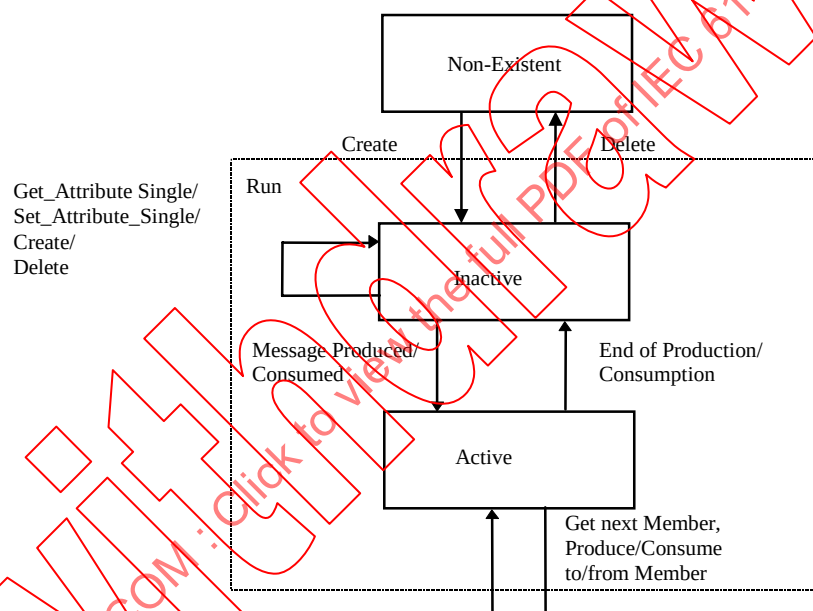
**Figure 27 – Dynamic Assembly State Transition Diagram**

Table 80 – Dynamic Assembly State Event Matrix

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

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

8.2.1.2.3.5 Specific requirements for a connection to the Assembly object

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

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

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

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

- configuration,
- producer,
- consumer,

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

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

8.2.1.2.4 Message Router Formal Model

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

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

The first part of the message (path information) shall then be parsed to determine the target object class. Interpretation of the class instance shall be performed on every service received by the Message Router. Any class instance that cannot be interpreted by the implementation of a Message Router in a device shall cause an error response to be returned with the "Object Not Found" status.

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

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

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

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

SYSTEM MANAGEMENT SERVICES:

- 1 (o) Mgt Service: Get_Attribute_All
- 3 (o) Mgt Service: Get_Attribute_List
- 10 (*) Mgt Service: Get_Attribute_Single

OBJECT MANAGEMENT SERVICES:

- 1 (o) Ops Service: Get_Attribute_All
- 3 (o) Ops Service: Get_Attribute_List
- 10 (*) Ops Service: Get_Attribute_Single

8.2.1.2.4.1 System Management Attributes (Class Attributes)

No specific requirement for this object.

8.2.1.2.4.2 Object Management Attributes

Object_List

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

This instance attribute has an access rule of Get only.

Number

Number of supported classes in the Classes attribute below.

Classes

List of class codes supported by the device

Number available

Maximum number of connections supported by the message router

This instance attribute has an access rule of Get only.

8.2.1.2.4.3 System Management (Class Level) and Object Management (Instance Level) Services

Get_Attribute_Single

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

8.2.1.2.4.4 Specific requirements of the Message Router object

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

The Message Router object shall support 1 or more connections to them. A Forward_Open with the parameters listed in Table 81 shall also be supported.

Table 81 – Message Router Object Forward_Open parameters

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

8.2.1.3 FAL Management Model ASE Service Specification

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

Get_Attribute_All
 Set_Attribute_All
 Get_Attribute_List
 Set_Attribute_List
 Reset
 Start
 Stop
 Create
 Delete
 Get_Attribute_Single
 Set_Attribute_Single
 Find_Next_Object_Instance

8.2.1.3.1 FAL Service Common Parameters

The following service parameters are common to all FAL services.

AREP

This parameter contains sufficient information to locally identify the entity to be used to convey the service : this entity could be local or the UCMM or a transport connection (AR). This parameter may use a dedicated identifier associated with a local entity, the identifier of a UCMM transaction record previously created, or the transport identifier returned by the connection establishment process and associated with the selected AR. The confirmation primitive should use the same value as the one sent in the corresponding request primitive.

Receiver/Server Local ID (id)

This parameter is locally generated by the Message Router ASE of the responding node, and identifies locally this invocation of the service. It is used to associate service responses with indications. Therefore, no two outstanding service invocations may be identified by the same ID (id) value.

Path

In the request, this parameter specifies the FAL APO or FAL APO element that is the destination of the service request. The path shall contain multiple segments which can indicate the network route to the node containing the FAL APO (in the case of multiple links), the identification of the APO element within the target node (Routing Path), and optional additional information for the target APO (Additional Path).

In the indication, this parameter shall contain only those segments beyond the logical class segment from the service request, i.e. the optional additional information for the target APO (AdditionalPath).

See IEC 61158-6 for more details on the Path structure and contents.

Port ID

This parameter shall indicate on which port of the device the service indication arrived.

Service status

This parameter provides information on the result of service execution. It is returned in all confirmed service response primitives (+ and -). It is composed of the following elements.

Status Code (mandatory)

This parameter shall indicate whether the service was successfully processed or not. If an error occurred, it indicates the type of error. For some errors, further identification of the error may be provided in the Extended Status parameter below. Available status codes are listed in 8.2.1.3.2.

Extended Status (conditional)

This conditional parameter shall further identify the error encountered when processing the request specific to the object being accessed. Actual usage of this Extended Status is dependant on the object class or service, but its format is fixed for each status code. When used, the values submitted in the response primitive are delivered unchanged in the confirmation primitive. Available extended status codes are described in 8.2.1.3.2.

8.2.1.3.2 FAL Service Status Codes

Table 82 specifies the range of possible status codes, and indicates whether an associated Extended Status is defined.

Table 82 –Status codes

Name	Description and meaning of status code	Extended Status
Success	Service was successfully performed by the object specified.	None
Connection failure	A connection related service failed along the connection path.	Conditional
Resource unavailable	Resources needed for the object to perform the requested behaviour were unavailable. Further object specific information shall be supplied in the object specific status field of the response.	None
Invalid value in object specific data parameter of a service request	A portion of the data supplied as a object specific data parameter of a service was invalid. The verification of the data shall be specified in the object specification of the object reporting the error.	None
Path segment error	The path segment identifier or the segment syntax was not understood by the processing node. The word offset to the first segment of the path that is not understood shall be supplied in the first word of the object specific status field of the response. The offset shall be zero based and calculated from the first word following the path size in the message. Path processing shall stop when a path segment error is encountered.	Conditional
Path destination unknown	The path is referencing an object class, instance or structure element that is not known or is not contained in the processing node. The word offset to the first segment component that references something that is unknown or not present in the processing node shall be supplied in the first word of the object specific status field of the response. The offset shall be zero based and calculated from the first word following the path size in the message. Path processing shall stop when a path destination unknown error is encountered.	Conditional
Partial transfer	Only part of the expected data was transferred.	None
Connection lost	The messaging connection was lost.	None
Unimplemented service	The service requested was not implemented or defined for this class or object instance.	None
Invalid attribute value	The value of an attribute of the object or class is invalid. The object specific status shall report the attribute number and the status code of the first attribute refusing data.	Conditional
Attribute list error	An attribute in the Get_Attribute_List or Set_Attribute_List response has a non zero status.	None
Already in requested mode/state	The object is already in the mode/state being requested by the service. The object specific status shall report the current status of the object.	Conditional
Object is not able to perform service in its current mode/state	The object can not perform the requested service in its current mode/state. The object specific status shall report the current status of the object.	Conditional
Object already exists	The requested instance of object to be created already exists.	None
Attribute value not settable	The object attribute is not a settable attribute. The object specific status shall report the number of the attribute refusing data.	None
Access permission does not allow service	The access permissions do not allow the object to perform the service. The access permissions available to the object shall be reported in the extended status. See Table 83 for details concerning extended status.	None
Device's mode/state does not allow object to perform service	The device containing the object does not allow the object to perform the service in the current mode/state of the device. The object specific status shall report the device current status, e.g. a controller may have a key switch which when set to the "hard run" state causes Service Requests to several different objects to fail (i.e. program edits) causing this status code to be returned.	Conditional
Response data too large	The data to be transmitted in the response buffer is larger than the allocated response buffer; therefore, no data was transferred.	None
Fragmentation of a primitive value	The service specified an operation that is going to fragment a primitive data value, i.e. half a REAL data type.	None
Not enough data	The service did not supply enough data to perform the specified operation.	None
Undefined attribute	The attribute specified is not defined for the class or object.	None
Too much data	The service supplied more data than was expected (depending on the service and the object, the service might still be processed).	None
Object does not exist	The object specified does not exist in the device.	None
Service fragmentation sequence not currently in progress	The fragmentation sequence for this service is not currently active for this data.	None
No stored attribute data	The attribute data of this object was not saved prior to the requested service.	None
Store operation failure	The attribute data of this object was not saved due to some failure during the attempt.	Conditional

(continued on next page)

Routing failure, request packet too large for network	The service request packet was too large for transmission on a network in the path to the destination. The routing device was forced to abort the service.	None
Routing failure, response packet too large for network	The service response packet was too large for transmission on a network in the path from the destination. The routing device was forced to abort the service.	None
Missing attribute list entry data	The service did not supply an attribute in a list of attributes that was needed by the service to perform the requested behaviour.	None
Invalid attribute value list	The service is returning the list of attributes supplied with status information for those attributes that were invalid.	Conditional
Embedded service error	An embedded service resulted in an error.	None
Connection Related Failure	A service failed because of an error condition related to the processing of a connection related service. This might occur during connected and unconnected messaging. The same extended status codes used for General Status Code 0x01 are returned for this error's extended status.	None
Write once value or medium already written	An attempt was made to write to a write once medium (e.g. WORM drive, PROM) that has already been written, or to modify a value that cannot be changed once established.	None
Invalid Response Received	An invalid response is received (e.g. response service code does not match the request service code, or response message is shorter than the minimum expected response size). This status code can serve for other causes of invalid replies.	None
Key Failure in path	The Key Segment which was included as the first segment in the path does not match the destination module. The object specific status shall indicate which part of the key check failed.	Conditional
Path Size Invalid	The size of the path which was sent with the Service Request is either not large enough to allow the Request to be routed to an object or too much routing data was included.	None
Unexpected attribute in list	An attempt was made to set an attribute that cannot be set at this time.	None
Invalid Member ID	The Member ID specified in the request does not exist in the specified Class/Instance/Attribute	None
Member not settable	A request to modify a non-modifiable member was received.	None
Reserved object/class specific	This range of status codes has been reserved for use by object and class specific services.	Conditional

Valid Extended Status formats are specified in Table 83.

Table 83 – Format of Extended Status

Name	Format of extended status	Meaning of Extended Status
Connection failure	extended_error_code	Additional error code specific to the connection process. List of additional codes is provided in IEC 61158-6.
Path segment error	path_segment_offset	Word offset from the beginning of the path of the incorrect segment
Path destination unknown	path_segment_offset	Word offset from the beginning of the path the first segment that is referencing a non-existent object or class
Invalid attribute value	attribute_number attribute_status	Number of the attribute refusing the data value status code explaining reason for invalid attribute value
Already in requested mode/state	object_status	Status/state/mode information of the object
Object is not able to perform service in its current mode/state	object_status	Status/state/mode information of the object
Device's mode/state does not allow object to perform service	device_status	Status/state/mode information of the device
Store operation failure	storage_status	Status/state/mode information of the storage media
Invalid attribute value list	attribute_count status_of_attributes [] attribute_number attribute_status	Number of attribute values being returned List of attribute status Number of attribute Status of attribute
Key Failure in path	key_element	Code indicating which part of the key check failed

8.2.1.3.3 Get_Attribute_All

The Get_Attribute_All service shall return the contents of all attributes of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

8.2.1.3.3.1 Service Primitives

The service parameters for this service are shown in Table 84.

Table 84 – Get_Attribute_All Service Parameters

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

Argument

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Attribute Data

This parameter returns a stream of information containing all of the attributes. APO classes which support this service shall define the format of this parameter. No fragmentation shall be allowed.

If supported, the structure of the Attribute Data information shall adhere to the Get_Attribute_All response structure as defined by the object class specification. The object class specification shall provide a detailed definition of the format of the data to be sent in the class and instance response messages.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

Path segment error (mandatory extended status, see Table 83)

Path destination unknown (mandatory extended status, see Table 83)

Connection lost (no extended status)

Unimplemented service (no extended status)

Response data too large (no extended status)

8.2.1.3.3.2 Service Procedure

No specific service procedure.

8.2.1.3.4 Set_Attribute_All

The Set_Attribute_All service shall modify the contents of the attributes of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

8.2.1.3.4.1 Service Primitives

The service parameters for this service are shown in Table 85.

Table 85 – Set_Attribute_All Service Parameters

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

Argument

The argument contains the parameters of the service request.

Attribute Data

This parameter specifies a stream of information containing all of the attributes. APO classes which support this service shall define the format of this parameter.

If supported, the structure of the Attribute Data information in the service request shall adhere to the Set_Attribute_All request structure as defined by the object class specification. The object class specification shall provide a detailed definition of the format of the data to be sent in the request message.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

Path segment error (mandatory extended status, see Table 83)

Path destination unknown (mandatory extended status, see Table 83)

Connection lost (no extended status)

Unimplemented service (no extended status)

Invalid attribute value (mandatory extended status, see Table 83)

Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)

8.2.1.3.4.2 Service Procedure

If the ability to modify an attribute changes based on the state of the object, the object specification shall specify when its attributes may be written.

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

8.2.1.3.5 Get_Attribute_List

The Get_Attribute_List service shall return the contents of the selected gettable attributes of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

8.2.1.3.5.1 Service Primitives

The service parameters for this service are shown in Table 86.

Table 86 – Get_Attribute_List Service Parameters

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

Argument

The argument contains the parameters of the service request.

Attribute Count

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

Attribute List

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Attribute Count

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

Attribute Data

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

Attribute Identifier

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

Attribute Status

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

Attribute Value

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

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

- Path segment error (mandatory extended status, see Table 83)
- Path destination unknown (mandatory extended status, see Table 83)
- Connection lost (no extended status)
- Unimplemented service (no extended status)
- Attribute list error (no extended status)

8.2.1.3.5.2 Service Procedure

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

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

8.2.1.3.6 Set_Attribute_List

The Set_Attribute_List service shall update the contents of the selected attributes of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

8.2.1.3.6.1 Service Primitives

The service parameters for this service are shown in Table 87.

Table 87 – Set_Attribute_List Service Parameters

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

Argument

The argument contains the parameters of the service request.

Attribute Count

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

Attribute Data

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

Attribute Identifier

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

Attribute Value

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Attribute Count

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

Attribute Status List

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

Attribute Identifier

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

Attribute Status

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

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

Path segment error (mandatory extended status, see Table 83)

Path destination unknown (mandatory extended status, see Table 83)

Connection lost (no extended status)

Unimplemented service (no extended status)

Attribute list error (no extended status)

Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)

Missing attribute list entry data

8.2.1.3.6.2 Service Procedure

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

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

8.2.1.3.7 Reset

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

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

8.2.1.3.7.1 Service Primitives

The service parameters for this service are shown in Table 88.

Table 88 – Reset Service Parameters

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

Argument

The argument contains the parameters of the service request.

Object Specific Data

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Object Specific Data

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

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

- Invalid value in object specific data parameter of a service request (no extended status)
- Path segment error (mandatory extended status, see Table 83)
- Path destination unknown (mandatory extended status, see Table 83)
- Connection lost (no extended status)
- Unimplemented service (no extended status)
- Invalid attribute value (mandatory extended status, see Table 83)
- Object is not able to perform service in its current mode/state (mandatory extended status, see Table 83)
- Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)

8.2.1.3.7.2 Service Procedure

If the execution of the Reset service request would place the object/device in a state that enables it to respond to the requester, then the response shall not be returned until the service has completed. If the execution of the Reset service would place the object/device in a state in which it is not able to respond, the object/device shall respond prior to executing the Reset service.

8.2.1.3.8 Start

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

NOTE Typically this would place an object into a running state/mode.

8.2.1.3.8.1 Service Primitives

The service parameters for this service are shown in Table 89.

Table 89 – Start Service Parameters

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

Argument

The argument contains the parameters of the service request.

Object Specific Data

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Object Specific Data

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

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

Path segment error (mandatory extended status, see Table 83)

Path destination unknown (mandatory extended status, see Table 83)

Connection lost (no extended status)

Unimplemented service (no extended status)

Invalid attribute value (mandatory extended status, see Table 83)

Already in requested mode/state (mandatory extended status, see Table 83)

Object is not able to perform service in its current mode/state (mandatory extended status, see Table 83)

Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)

8.2.1.3.8.2 Service Procedure

If supported by an object class or instance, the effect of this service shall be as documented in that object specification.

8.2.1.3.9 Stop

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

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

8.2.1.3.9.1 Service Primitives

The service parameters for this service are shown in Table 90.

Table 90 – Stop Service Parameters

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

Argument

The argument contains the parameters of the service request.

Object Specific Data

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Object Specific Data

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

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

- Path segment error (mandatory extended status, see Table 83)
- Path destination unknown (mandatory extended status, see Table 83)
- Connection lost (no extended status)
- Unimplemented service (no extended status)
- Invalid attribute value (mandatory extended status, see Table 83)
- Already in requested mode/state (mandatory extended status, see Table 83)
- Object is not able to perform service in its current mode/state (mandatory extended status, see Table 83)
- Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)

8.2.1.3.9.2 Service Procedure

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

8.2.1.3.10 Create

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

8.2.1.3.10.1 Service Primitives

The service parameters for this service are shown in Table 91.

Table 91 – Create Service Parameters

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

Argument

The argument contains the parameters of the service request.

Object Specific Data

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Instance ID

This mandatory parameter indicates the integer value assigned to identify the newly created object instance.

Object Specific Data

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

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

- Path segment error (mandatory extended status, see Table 83)
- Path destination unknown (mandatory extended status, see Table 83)
- Connection lost (no extended status)
- Unimplemented service (no extended status)
- Invalid attribute value (mandatory extended status, see Table 83)
- Already in requested mode/state (mandatory extended status, see Table 83)
- Object is not able to perform service in its current mode/state (mandatory extended status, see Table 83)
- Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)
- Missing attribute list entry data (no extended status)
- Invalid attribute value list (mandatory extended status, see Table 83)

8.2.1.3.10.2 Service Procedure

If this service is supported, the object instance shall be created and initialised in accordance with the object specification.

Any error shall result in the object instance not being created.

8.2.1.3.11 Delete

The Delete service shall delete an object instance of the specified object class.

8.2.1.3.11.1 Service Primitives

The service parameters for this service are shown in Table 92.

Table 92 – Delete Service Parameters

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

Argument

The argument contains the parameters of the service request.

Object Specific Data

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Object Specific Data

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

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

Path segment error (mandatory extended status, see Table 83)

Path destination unknown (mandatory extended status, see Table 83)

Connection lost (no extended status)

Unimplemented service (no extended status)

Object is not able to perform service in its current mode/state (mandatory extended status, see Table 83)

Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)

8.2.1.3.11.2 Service Procedure

All resources shall be deallocated and returned to the system.

8.2.1.3.12 Get_Attribute_Single

The Get_Attribute_Single service shall return the contents of the specified attribute of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

8.2.1.3.12.1 Service Primitives

The service parameters for this service are shown in Table 93.

Table 93 – Get_Attribute_Single Service Parameters

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

Argument

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

Path

This parameter shall include a path segment specifying the attribute number.

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Attribute Data

This parameter contains the requested attribute data.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

Connection lost (no extended status)

Unimplemented service (no extended status)

Invalid attribute value (mandatory extended status, see Table 83)

Attribute value not settable (no extended status)

Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)

Object does not exist (no extended status)

8.2.1.3.12.2 Service Procedure

No specific service procedure.

8.2.1.3.13 Set_Attribute_Single

The Set_Attribute_Single service shall modify the contents of the specified attribute of the specified object class (APO System Management attributes) or instance (APO Object Management attributes).

8.2.1.3.13.1 Service Primitives

The service parameters for this service are shown in Table 94.

Table 94 – Set_Attribute_Single Service Parameters

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

Argument

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

Path

This parameter shall include a path segment specifying the attribute number.

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

Attribute Data

This parameter contains the value to which the specified attribute is to be modified. The attribute data shall be validated prior to the modification taking place.

Result(+)

This selection type parameter indicates that the service request succeeded.

Service status

This parameter must indicate Success.

Object Specific Data

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

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

- Path segment error (mandatory extended status, see Table 83)
- Path destination unknown (mandatory extended status, see Table 83)
- Connection lost (no extended status)
- Unimplemented service (no extended status)
- Invalid attribute value (mandatory extended status, see Table 83)
- Attribute value not settable (no extended status)
- Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)
- Object does not exist (no extended status)

8.2.1.3.13.2 Service Procedure

No specific service procedure.

8.2.1.3.14 Find_Next_Object_Instance

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

8.2.1.3.14.1 Service Primitives

The service parameters for this service are shown in Table 95.

Table 95 – Find_Next_Object_Instance Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Path size	M	M(=)		
Path	M	M(=)		
Port ID		M		
Maximum Returned Values	M	M(=)		
Result (+)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)
Number of List Members			M	M(=)
Instance ID List			M	M(=)
Result (-)			S	S(=)
AREP				M
Receiver/Server Local ID			M	
Service status			M	M(=)

Argument

The argument contains the parameters of the service request.

Path

This parameter shall include a path segment specifying an Instance ID, which will be used to determine the starting point for the search.

Maximum Returned Values

This parameter indicates the maximum number of Instance ID values to be returned in the response message.

Result(+)

This selection type parameter indicates that the service request succeeded.

Number of List Members

This parameter contains the actual number of Instance ID's returned in this response message. This number of Instance ID's shall be less than or equal to the maximum specified in the request.

Instance ID List

This parameter contains the list of returned Instance ID's.

Result(-)

This selection type parameter indicates that the service request failed.

Service status

This parameter must indicate an error among the following choices :

- Path segment error (mandatory extended status, see Table 83)
- Path destination unknown (mandatory extended status, see Table 83)
- Connection lost (no extended status)
- Unimplemented service (no extended status)
- Object is not able to perform service in its current mode/state
- Device's mode/state does not allow object to perform service (mandatory extended status, see Table 83)

8.2.1.3.14.2 Service Procedure

The object class shall utilise the value specified in the Instance ID of the request message to determine the starting point for the search as described below:

1. If the Instance ID in the request message is zero (0), then the class shall start with the numerically lowest Instance ID;
2. If the Instance ID in the request message is not zero (0), then the class shall start with the next Instance ID whose value is numerically greater than the specified Instance ID.
3. If the Instance ID in the request message is greater than or equal to the numerically highest Instance ID within the class, then the value zero (0) shall be returned as Instance ID.

The class shall return a list of Instance IDs associated with existing objects, beginning at the starting point and continuing in ascending Instance ID value order. It shall return Instance ID value zero (0) to indicate that the end of the list has been reached.

Figure 28 provides a general example of this service.

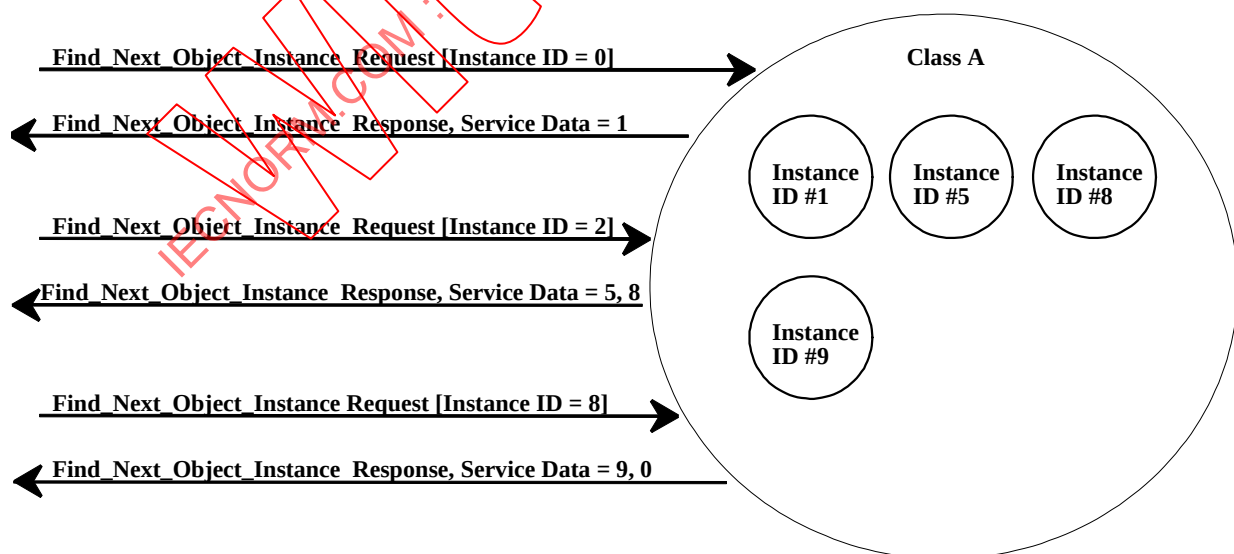


Figure 28 – Example of Find_Next_Object_Instance service

8.2.2 Connection Manager ASE

8.2.2.1 Overview

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

8.2.2.2 Connection Manager Class Specification

8.2.2.2.1 Connection Manager Formal Model

FAL ASE: **FAL Management ASE**
CLASS: Connection_Manager_Object
CLASS ID: <td>
PARENT CLASS: Base_Object

ACCESS ATTRIBUTES:

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

SYSTEM MANAGEMENT ATTRIBUTES (CLASS ATTRIBUTES) :

- 1 (o) Attribute: Revision = 1
- 2 (o) Attribute: Max Instance
- 4 (o) Attribute: Optional attribute list
- 4.1 (m) Attribute: Number of attributes
- 4.2 (m) Attribute: Optional attributes

OBJECT MANAGEMENT ATTRIBUTES (INSTANCE ATTRIBUTES) :

- 1 (o) Attribute: OpenReqs
- 2 (o) Attribute: OpenFormat Rejects
- 3 (o) Attribute: OpenResource Rejects
- 4 (o) Attribute: OpenOther Rejects
- 5 (o) Attribute: CloseReqs
- 6 (o) Attribute: CloseFormat Rejects
- 7 (o) Attribute: CloseOther Rejects
- 8 (o) Attribute: ConnTimeouts
- 9 (o) Attribute: Connection Entry List
- 9.1 (m) Attribute: NumConnEntries
- 9.2 (m) Attribute: ConnOpenBits
- 11 (o) Attribute: CpuUtilization
- 12 (o) Attribute: MaxBuffSize
- 13 (o) Attribute: BufSize Remaining

SYSTEM MANAGEMENT SERVICES:

- 1 (o) Mgt Service: Get_Attribute_All
- 3 (o) Mgt Service: Get_Attribute_List
- 10 (o) Mgt Service: Get_Attribute_Single

OBJECT MANAGEMENT SERVICES:

- 1 (o) Ops Service: Get_Attribute_All
- 2 (o) Ops Service: Set_Attribute_All
- 3 (o) Ops Service: Get_Attribute_List
- 4 (o) Ops Service: Set_Attribute_List
- 10 (o) Ops Service: Get_Attribute_Single
- 11 (o) Ops Service: Set_Attribute_Single

OBJECT SPECIFIC SERVICES:

- 1 (m) OpsService: CM_Open
- 2 (m) OpsService: CM_Close
- 3 (o) OpsService: CM_Unconnected_Send

8.2.2.2.1.1 System Management Attributes (Class Attributes)

No specific requirement for this object.

8.2.2.2.1.2 Object Management Attributes

OpenReqs

Number of Open requests received including Null Open requests.

This instance attribute has an access rule of Set.

OpenFormat Rejects

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

This instance attribute has an access rule of Set.

OpenResource Rejects

Number of Open requests rejected by this node.

This instance attribute has an access rule of Set.

OpenOther Rejects

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

This instance attribute has an access rule of Set.

CloseReqs

Number of Close requests received.

This instance attribute has an access rule of Set.

CloseFormat Rejects

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

This instance attribute has an access rule of Set.

CloseOther Rejects

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

This instance attribute has an access rule of Set.

ConnTimeouts

Number of connections which have been timed out by this node after they were opened.

This instance attribute has an access rule of Set.

Connection Entry List

List of connections.

This instance attribute has an access rule of Get only.

NumConnEntries

Number of entries in the ConnOpenBits Attribute.

ConnOpenBits

List of connection data which may be individually queried. Each entry represents a possible connection, and has one of two states : No Connection or Connection Established. More information on a connection can be obtained using a dedicated service.

CpuUtilization

CPU utilization in tenths of a percent.

This instance attribute has an access rule of Get only.

MaxBuffSize

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

This instance attribute has an access rule of Get only.

BufSize Remaining

Amount of buffer space available at this time (buffer size is in bytes).

This instance attribute has an access rule of Get only.

8.2.2.2.1.3 System Management (Class Level) and Object Management (Instance Level) Services

No specific requirement for this object.

8.2.2.2.1.4 Object Specific Services

These Connection Manager services are available only through the Unconnected Message Manager AR (UCMM). They are used either to establish/de-establish an application connection, or to send messages through routers without an established connection.

CM_Open

This service is used by connection originator users to establish an application connection.

CM_Close

This service is used by connection originator users to de-establish an application connection.

CM_Unconnected_Send

The CM_Unconnected_Send service shall allow an application to send a message to a device through multiple links without first setting up a connection. This optional service is mandatory for originator devices and devices that route messages between links.

8.2.2.3 Connection Manager ASE Service Specification

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

CM_Open

CM_Close

CM_Unconnected_Send

8.2.2.3.1 Connection Manager Common Service Parameters

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

NOTE Complete specification of these parameters is provided in IEC 61158-6.

Originator Local ID

This parameter is locally generated by the originator node, and identifies locally this invocation of the service. It is used to associate service confirmations with requests. Therefore, no two outstanding service invocations may be identified by the same ID value.

Target Local ID

This parameter is locally generated by the target node, and identifies locally this invocation of the service. It is used to associate service responses with indications. Therefore, no two outstanding service invocations may be identified by the same ID value.

Path

This parameter specifies the FAL APO or FAL APO element that is the target of a connection establishment (or de-establishment) request, or an unconnected message request. The path shall contain multiple segments which can indicate the network route to the node containing the FAL APO (in the case of multiple links), the identification of the APO element within the target node (Routing Path), and optional additional information for the target APO (Additional Path). This Additional Path parameter is passed on to the target application, and could be a data segment used to configure the application, the transport class, and the trigger.

Transport identifier

The Transport identifier parameter is generated locally by the originator Connection Manager, and shall serve as a further reference for the user to the newly established connection.

Transaction_priority

The Transaction_priority parameter shall determine which Data Link Layer priority to use for the messages that convey the service requests and responses, and shall be one of LOW or HIGH.

NOTE The definition of DLL priority is specified in IEC 61158-4.

Transaction_timeout

The Transaction_timeout parameter shall determine the time to wait for service completion. The units for Transaction_timeout shall be milliseconds.

O2T_ConnParm / T2O_ConnParm

The O2T parameter shall specify the characteristics of the originator to target ($O \Rightarrow T$) communication on the new connection. The T2O parameter shall specify the characteristics of the target to originator ($T \Rightarrow O$) communication.

CM_RPI

The Requested Packet Interval (CM_RPI) shall specify the expected time between packets, i.e. how frequently the originating application requires the transmission of data from the target application. It shall be expressed as a number of microseconds.

Type

The Type parameter shall indicate the type of the connection, and shall be one of NULL, MULTICAST, or POINT2POINT. A value of NULL indicates that the specified direction ($O \Rightarrow T$ or $T \Rightarrow O$) shall have no transport associated. A value of MULTICAST indicates that other connections may later participate in the transport. Conversely, a value of POINT2POINT indicates that this connection shall not allow any other transport to participate.

Priority

The Priority parameter shall determine which Data Link Layer priority to use for the new transports, and shall be one of LOW, HIGH, or SCHEDULED.

Variable

The Variable parameter specifies whether every application data which traverses the new transport shall be the same size or not, and shall be either TRUE (variable size) or FALSE (fixed size).

Size

The Size parameter shall specify the size (in bytes) of the largest application data packet for this connection.

CM_RPI_multiplier

The product of the CM_RPI_multiplier parameter and the actual packet interval (CM_API) shall specify the time-out for the transport. If the application has not used the transport within this time-out, the transport shall close, which shall subsequently close the connection.

Trigger

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

NOTE This parameters shall be used to configure the transport(s). Its meaning to the transports is described in 8.3.1.

Trans_class

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

NOTE This parameters shall be used to configure the transport(s). Its meaning to the transports is described in 8.3.1.

Is_server

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

NOTE This parameters shall be used to configure the transport(s). Its meaning to the transports is described in 8.3.1.

Manufacturer_ID

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

Originator_Ser_Num

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

Connection Serial Number

The Connection Serial Number parameter is a value selected by the originator Connection Manager to uniquely identify a connection within the originator device.

Service status

This parameter provides information on the result of service execution. It is returned in all Connection Manager service response/confirmation primitives. It has the same definition as the Service Status parameter of the FAL Management services. Corresponding available Status Codes and Extended Status formats are detailed in 8.2.2.3.2.

O2T_CM_API / T2O_CM_API

The O2T_CM_API and T2O_CM_API shall specify the actual packet interval that shall be used for the new connection, i.e. how frequently the connection produces its data. It shall be expressed in microseconds. These values may be different than the requested packet intervals, but shall always be equal or smaller than the requested interval.

Remaining Path Size

In the failure response, the remaining_path parameter specifies the “pre-stripped” size. This is the size of the path when the node first receives the request and has not yet started processing it. A target node may instead return 0 (zero) for this parameter. Associated with the Service Status (Status Code and Extended Status), this parameter allows to identify the type and location of any errors found during connection establishment or de-establishment.

Response Data

If the target has any application specific response data, it shall be returned in the Response Data parameter.

8.2.2.3.2 Connection Manager Service Status Codes

Specific status codes for the Connection Manager are detailed in IEC 61158-6.

8.2.2.3.3 CM_Open

The CM_Open service shall be used by connection originator users to establish an application connection to a specified target object instance or instance element. The CM_Open request contains the parameters to select connection types, transports and to specify the requested packet interval in both the originator to target and target to originator direction.

8.2.2.3.3.1 Service Primitives

The service parameters for this service are shown in Table 96.

Table 96 – CM_Open Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
Originator Local ID	M			
Target Local ID		M		
Path	M			
Routing Path	M			
Additional Path	U	U(=)		
Transport identifier	M			
Transaction_priority	M			
Transaction_timeout	M			
O2T_ConnParm	M	M(=)		
CM_RPI	M	M(=)		
Type	M	M(=)		
Priority	M	M(=)		
Variable	M	M(=)		
Size	M	M(=)		
T2O_ConnParm	M	M(=)		
CM_RPI	M	M(=)		
Type	M	M(=)		
Priority	M	M(=)		
Variable	M	M(=)		
Size	M	M(=)		
CM_RPI_multiplier	M	M(=)		
Trigger	M	M(=)		
Trans_class	M	M(=)		
Is_server	M	M(=)		
Manufacturer_ID		M		
Originator_Ser_Num		M		
Connection Serial Number		M		
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Transport identifier				M
O2T_CM_API			M	M(=)
T2O_CM_API			M	M(=)
Remaining Path Size			M	M(=)
Response Data			U	U(=)

Argument

The argument contains the parameters of the service request.

Transport identifier

The transport identifier parameter in the request shall allow a previously opened connection to be reused (this is the same identifier as the one returned in the CM_Open confirmation for this first connection). If Transport identifier is zero, then a new transport shall be created.

NOTE Reusing a transport allows an originator to establish multicast connections in the O⇒T direction.

Result

This parameter indicates whether the service request succeeded or failed.

8.2.2.3.2 Service Procedure

The CM_Open request primitive, sent from the originating application, first triggers the connection establishment process into the local Connection Manager, before being forwarded to the remote target Connection Manager.

The corresponding CM_open indication primitive shall then be sent to the target application from the target Connection Manager, after some local processing. The application shall reply to the CM_open indication primitive with a CM_open response whether the application accepts the connection or not.

After the connection has been established, or if the connection attempt failed, a CM_open confirm primitive shall be sent from the local originator connection manager back to the originating application. The confirmation shall indicate the status of the connection and shall provide information about the connection including the Transport Identifier parameter, which shall serve as a reference to the newly established connection.

8.2.2.3.4 CM_Close

The CM_Close service shall be used by connection originator users to de-establish ("close") an application connection previously established to a target object instance or instance element, and to de-allocate the resources associated with the connection. The service request shall indicate the connection to close by using the original path and the Transport identifier. The originating application shall save both the connection path required to establish the connection, and the Transport identifier returned by the CM_open confirmation.

8.2.2.3.4.1 Service Primitives

The service parameters for this service are shown in Table 97.

Table 97 – CM_Close Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M			
Originator Local ID	M			
Target Local ID		M		
Path	M			
Routing Path	M			
Additional Path	U	U(=)		
Transport identifier	M			
Transaction_priority	M			
Transaction_timeout	M			
Manufacturer_ID		M		
Originator_Ser_Num		M		
Connection Serial Number		M		
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Transport identifier				M
Remaining Path Size			M	M(=)
Response Data			U	U(=)

Argument

The argument contains the parameters of the service request.

Result

This parameter indicates whether the service request succeeded or failed.

8.2.2.3.4.2 Service Procedure

The CM_close request primitive, sent from the originator application, first triggers the connection de-establishment process into the local Connection Manager, before being forwarded to the remote target Connection Manager. of a connection to de-establish the connection and de-allocate the resources associated with the connection. The service shall be sent from the originator application to the local connection manager.

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

The CM_close response shall be sent from the target application to the target Connection Manager to indicate the success of the CM_close request. A successful CM_close response primitive shall be returned when the de-establishment has been acknowledged by the target application. Once the response indicating success has been received, the connection shall be

closed. The target application may optionally pass back application specific information in the close response. This application response information shall be object specific.

A target application shall not refuse to close a connection but may not be able to find the specified connection. Once the CM_close_response is received, the specified connection shall be closed or shall not be present. The object specific status words shall contain information about the reason for the failure. The remaining connection path size shall contain the path size from the point at which the close connection failed.

The CM_close confirmation shall be sent back from the local Connection Manager to the originating application to indicate the status of the close connection request.

A successful CM_close confirmation shall be returned when the close has been acknowledged by all the nodes involved in the original connection. The confirmation shall indicate the Transport identifier of the closed connection. Once the confirmation indicating success has been received, the connection shall be closed. The target application may optionally pass back application specific information in the close response. This information shall be object specific.

In case of a failure response, the Service status parameter shall contain information about the reason for the failure. The Remaining Path Size shall contain the path size from the point at which the close connection failed. This Remaining Path Size shall be the "pre-stripped" size. This shall be the size of the path when the node first receives the request and has not yet started processing it. A target node may return either the "pre-stripped" size or 0 for the Remaining Path Size.

8.2.2.3.5 CM_Unconnected_Send

The CM_Unconnected_Send service shall allow an application to send a message to a device through multiple links without first setting up a connection. This optional service is mandatory for originator devices and devices that route messages between links.

8.2.2.3.5.1 Service Primitives

The service parameters for this service are shown in Table 98.

Table 98 – CM_Unconnected_Send Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M		
Originator Local ID	M			
Target Local ID		M		
Path	M			
Routing Path	M			
Additional Path	U	U(=)		
Transaction_priority	M			
Transaction_timeout	M			
OM_Service Code	M	M(=)		
OM_Service Request Parameters	M	M(=)		
Result				
Originator Local ID				M
Target Local ID			M	
Service status			M	M(=)
Remaining Path Size			M	M(=)
OM_Service Response Parameters			M	M(=)

Argument

The argument contains the parameters of the service request.

OM_Service Code

This parameter indicates the code of the Object Management service to be performed by the target element.

OM_Service Request Parameters

This parameter contains the specific arguments parameters of the Object Management service to be performed by the target element.

Result

This parameter indicates whether the service request succeeded or failed.

OM_Service Response Parameters

This parameter contains the specific result parameters of the Object Management service to be performed by the target element.

8.2.2.3.5.2 Service Procedure

The CM_Unconnected_Send service shall use the Connection Manager object in each intermediate node to forward the message and to remember the return path. The UCMM of each link shall be used to forward the request from Connection Manager to Connection Manager just as it is for the Forward_Open service; however, no connection shall be built. The

CM_Unconnected_Send service shall be sent to the local Connection Manager and shall be sent between intermediate nodes. When an intermediate node removes the last port segment, the message shall be formatted as a UCMM message and sent to the port and link address of the last segment.

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

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

8.3 Type 2 AR's

8.3.1 Overview

8.3.1.1 General

One of the roles of the Type 2 Application Layer is to establish and maintain connections. A connection is similar to a telephone circuit. When a call is placed, the telephone system selects a path for your call and sets up each switching station in the route to handle it. As long as the call continues, the resulting virtual circuit remains open, carrying data or voice traffic. In the telephone system, a single call can travel over multiple, different type links, and the format of the data can change from link to link, but the connection appears the same to both the caller and the party called. Sound in one end becomes sound out the other.

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

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

The unconnected service is provided by the Unconnected Message Manager (UCMM). Messages received through the UCMM are forwarded to the Message Router (MR), which direct them to the appropriate internal object for execution. Connections are established by specific unconnected messages sent through the Connection Manager (CM) using UCMM services. Connections may be established either to the Message Router (for messaging purpose), or directly to an application object. Connection target is specified using a connection path, and may be either on the local or a remote link, through several intermediate router nodes. Once a connection is established with an application object, the UCMM, MR and CM are no longer required, since data will be exchanged directly with the connected objet, based on the corresponding connection ID. Connected messages sent to the Message Router will be forwarded to the appropriate internal object for execution.

The connected and unconnected protocols specified within this IEC 61158-5 are collectively known as the control and information protocol (CIP).

8.3.1.2 Message Router

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

8.3.1.3 Application connections

8.3.1.3.1 Connections

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

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

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

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

8.3.1.3.2 Production Trigger, Transport Class and CM_RPI

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

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

The triggers are used in conjunction with the transport class and CM_RPI to control when data are sampled and sent on the link. Table 99 shows how these factors relate to each other in determining when data are produced.

Table 99 – How production trigger, transport class, and CM_RPI determine when data is produced

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

8.3.1.3.3 Polling

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

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

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

8.3.1.4 Transport connections

8.3.1.4.1 General

A transport connection is a logical binding between two transport instances that is made to enable two applications to transfer data over network connections. A transport instance is a specific implementation of a transport function. Every transport connection is built on one or two network

connections. Figure 29 uses the communication model to show the context in which transport connections are made.

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

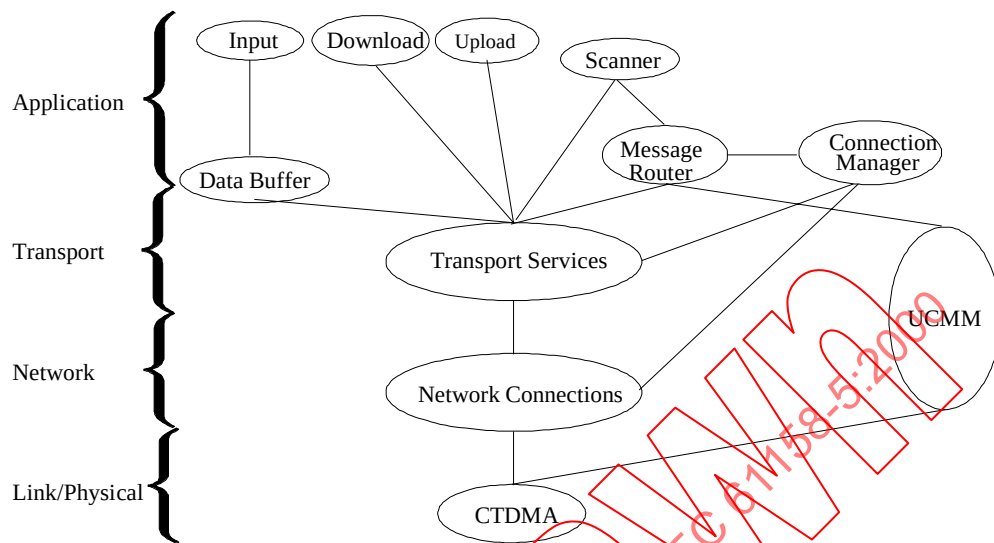


Figure 29 – Context of Transport Services within the Connection Model

In Figure 30, application A interfaces to a client transport instance to trigger the production of data over the network connection, and application B interfaces to a server transport instance to be notified that data has been written to the transport protocol data unit (T-PDU) buffer. A T-PDU comprises one packet of data as well as its link, network, and transport headers. A T-PDU buffer is a memory location in which one T-PDU can be stored.

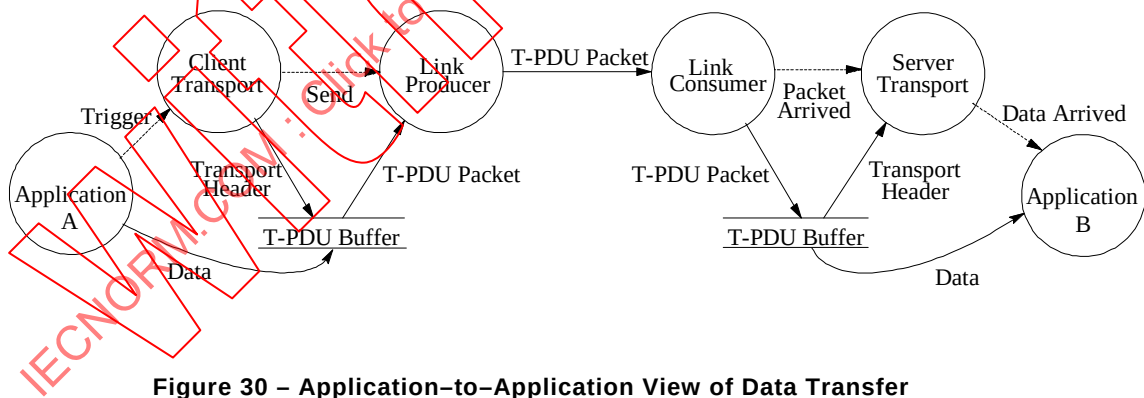


Figure 30 – Application-to-Application View of Data Transfer

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

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

8.3.1.4.2 Components of Transport Connections

8.3.1.4.2.1 Components

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

8.3.1.4.2.2 Network Connections

A logical binding between a producer and consumer shall form a network connection. The Connection Manager shall establish this connection. When a producer wants to send a message, it shall provide just the production ID. The production ID shall imply the path that the connection manager specified. Because of this implied route, no addressing or routing information other than the production ID is used at run time.

There is no queuing of data along a network connection. Overwrites shall be permitted at all intermediate nodes. Those applications that require queuing shall use an appropriate transport class to deliver this behaviour.

Link Producer

The link producer shall be responsible for fetching data from the data buffer and producing it on the data link after the send event has been received. Data buffer management schemes such as overwrite, double buffered, triple buffered are not excluded, but since they are implementation specific, they are not specified here.

Behaviour of the link producer is shown in Figure 31.

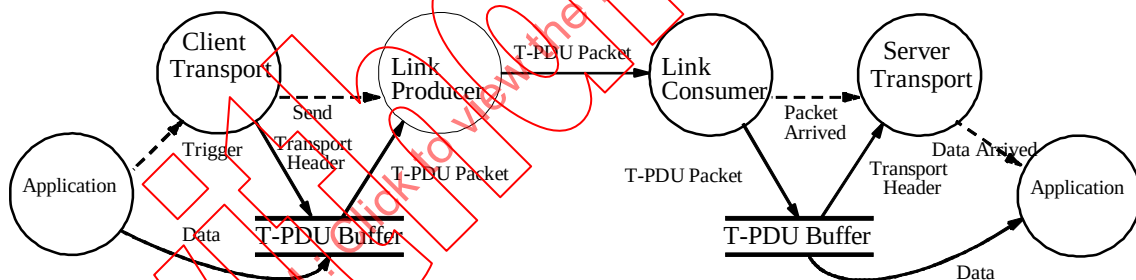


Figure 31 – Data Flow Diagram for a link producer

Class object description of the Link Producer is detailed below.

FAL ASE:	Link Producer ASE
CLASS:	Link_Producer
CLASS ID:	<td>
PARENT CLASS:	TOP
STATIC ATTRIBUTES:	
1	(m) Attribute: Production ID
2	(m) Attribute: Priority = SCHEDULED HIGH LOW
3	(m) Attribute: Connection Size
4	(m) Attribute: Destination Type = POINT2POINT MULTICAST
5	(m) Attribute: State
SERVICES:	
1	(m) OpsService: Create
2	(o) OpsService: Delete
3	(m) OpsService: Send

Create

Create instance of a link producer.

Delete

Delete instance of a link producer.

Send

Produce the T-PDU buffer.

Link Consumer

The link consumer is responsible for receiving T-PDU packets, storing them in the T-PDU buffer and notifying the server transport class that a packet has arrived. Data buffer management schemes such as overwrite, double buffered, triple buffered are not excluded, but since they are implementation specific, they are not specified here.

Behaviour of the link producer is shown in Figure 32.

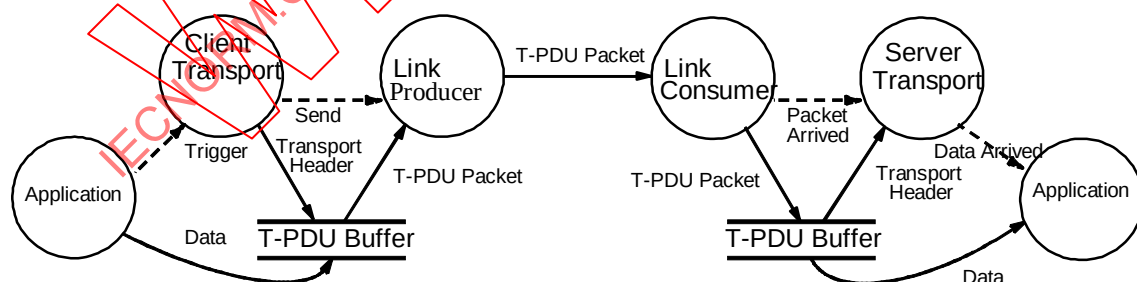


Figure 32 – Data Flow Diagram for a link consumer

Class object description of the Link Consumer is detailed below.

FAL ASE:	Link Consumer ASE
CLASS:	Link_Consumer
CLASS ID:	<tbid>
PARENT CLASS:	TOP
STATIC ATTRIBUTES:	
1	(m) Attribute: Consumption ID
2	(m) Attribute: Priority = SCHEDULED HIGH LOW
3	(m) Attribute: Connection Size
4	(m) Attribute: Destination Type = POINT2POINT MULTICAST
5	(m) Attribute: State
SERVICES:	
1	(m) OpService: Create
2	(o) OpService: Delete
3	(m) OpService: Send

Create

Create instance of a link consumer.

Delete

Delete instance of a link consumer.

8.3.1.4.2.3 Triggers

A trigger shall cause data to be sent. This trigger can be anything that makes sense from an application point of view, such as a clock on a module, a network trigger, a scheduler, an asynchronous event or a timer, as shown in Figure 33.

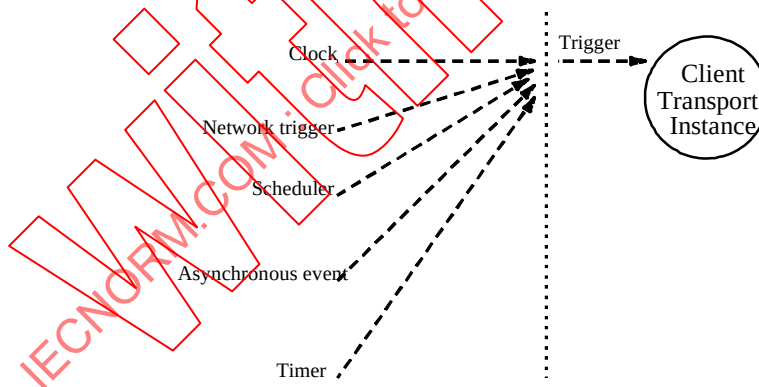


Figure 33 – Triggers

It is important to note that the writing of data to the buffer and triggering of that data are separate functions. An application may write to a buffer more often than it triggers data or trigger data more often than it writes to the buffer. Applications shall write data to the data buffer before it is initially triggered to ensure that valid data is sent.

8.3.1.4.3 Creating Transport Connections

After the application has bound the producer to the consumer at the network level of the communication model, the application creates the transport connection by binding the producer,

consumer, and T-PDU buffer(s) to the transport instances at the transport level. An application may support as many transport connections as resources permit.

The simplest transport functions use a single network connection, which can be either point-to-point or multicast; the more complex transport functions also use one or more point-to-point return network connections. The transport functions correspond to the transport classes. Each transport class provides a specified level of transport services.

8.3.1.4.4 Binding Transports to Producers and Consumers of Network Connections

The binding rules followed to create transport connections are as follows:

- A client transport instance is bound to exactly one producer instance;
- A client transport instance can be bound to zero, one, or many consumer instances;
- A server transport instance can be bound to zero or one producer instances;
- A server transport instance is bound to exactly one consumer instance;
- A server transport instance can be bound to a producer instance and a consumer instance, but can not be bound to a producer instance without also being bound to a consumer instance.

The following examples demonstrate application of the binding rules to different kinds of transport connections that include network connections with different destination types.

- For a point-to-point connection that does not have a reverse data path, the producer is bound to the client transport instance and the consumer is bound to the server transport instance as shown in Figure 34:

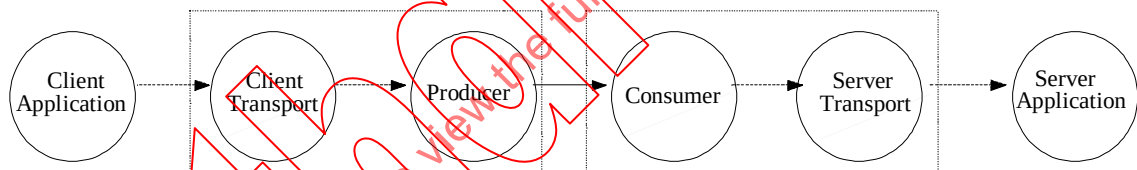


Figure 34 – Binding Transport Instances to the Producer and Consumer of a Transport Connection that Does Not Have a Reverse Data Path

- For a point-to-point connection that does have a reverse data path:
 - In the client-to-server direction, the producer is bound to the client transport instance, and the consumer is bound to the server transport instance.
 - In the server-to-client direction, the producer is bound to the server transport instance, and the consumer is bound to the client transport instance.

Figure 35 illustrates this scenario.

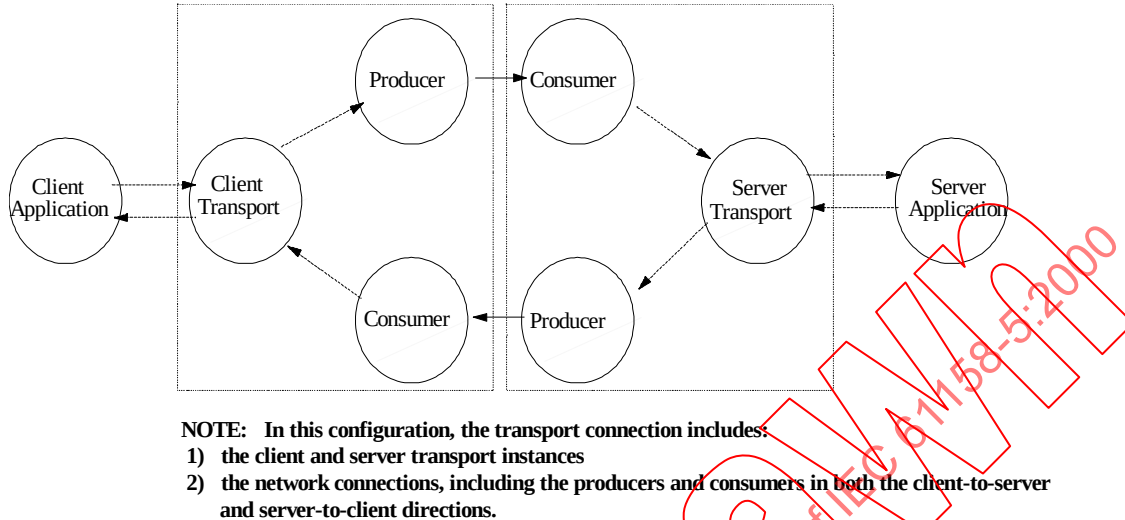


Figure 35 – Binding Transport Instances to the Producers and Consumers of a Transport Connection that Does Have a Reverse Data Path

- For a multicast connection that does not have a reverse data path, the producer is bound to the client transport instance, and each consumer is bound to a separate server transport instance. It is important to note that there is only one client transport instance, and there are as many server transport instances as there are client-to-server network connections.

Figure 36 illustrates this scenario.

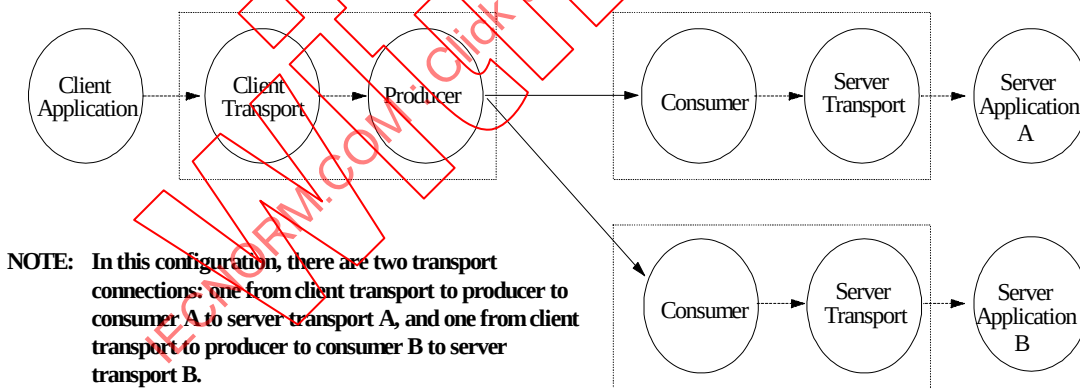


Figure 36 – Binding Transport Instances to the Producer and Consumers of a Multicast Connection When the Transport Connection Does Not Have a Reverse Data Path

- For a multicast connection that does have a reverse data path:
 - In the client-to-server direction, the producer is bound to the client transport instance, and each consumer is bound to a separate server transport instance.
 - In the server-to-client direction, each producer is bound to a separate server transport instance, and each consumer is bound to the client transport instance. It is important to note that there is only one client transport instance, and there are as many server transport instances as there are client-to-server network connections.

Figure 37 illustrates this scenario.

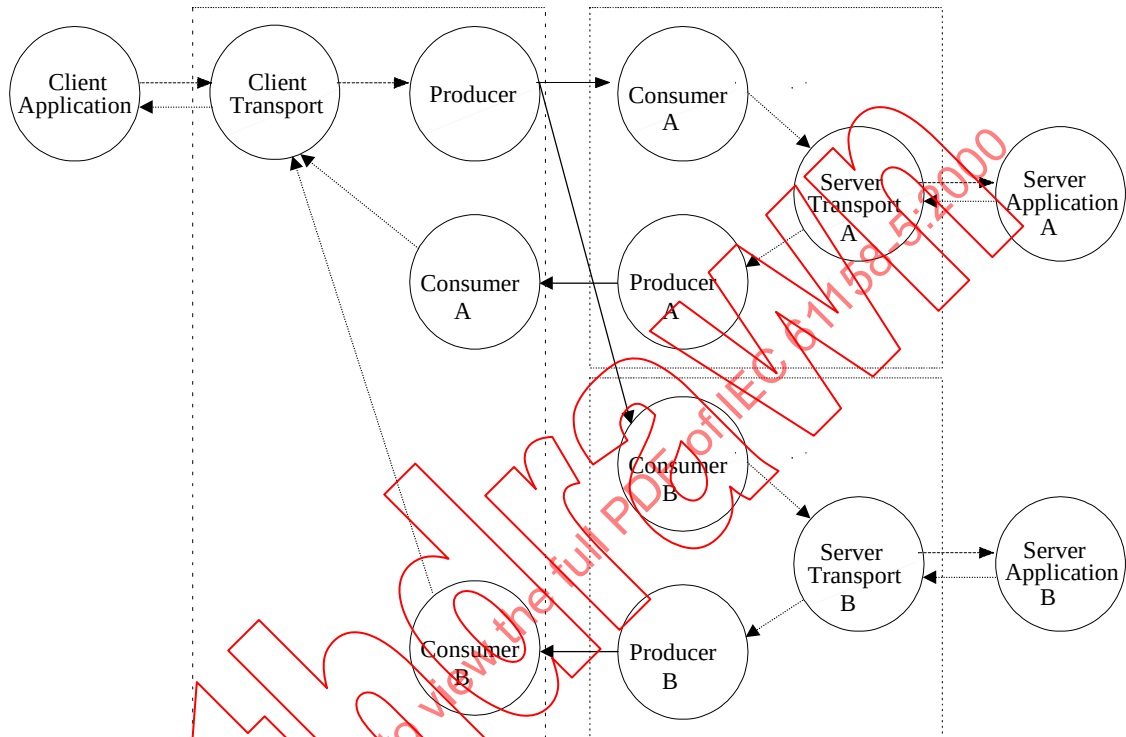


Figure 37 – Binding Transport Instances to the Producers and Consumers of a Multicast Connection When the Transport Connection Does Have Reverse Data Paths

8.3.1.4.5 Transport Type

8.3.1.4.5.1 General

The transport type shall determine the target behaviour concerning the relationship between different connections each sharing a producer, and shall be one of LISTEN_ONLY, INPUT_ONLY, or EXCLUSIVE_OWNER.

8.3.1.4.5.2 Listen Only

If a connection has a transport type of listen only, it shall be dependent on another connection for its existence. Devices that wish to listen to multicast data without providing configuration or scheduling information may use this type of transport. If the connection on which a listen only connection depends is closed or times out, the listen only connection shall also be closed.

Unlike exclusive owner connections, listen only connections need not have a method to specify the RUN/IDLE state of the originator.

8.3.1.4.5.3 Input Only

If a connection has a transport type of input only, it shall not be dependent on any other connection for its existence. For scheduled input only connection, the Forward_Open path shall contain a schedule segment. No O⇒T data shall be sent; therefore, a target may accept many input only connections. A specific implementation may limit the number of input only connections it accepts.

Unlike exclusive owner connections, input only connections shall not have a method to specify monitoring of the RUN/IDLE state of the originator.

8.3.1.4.5.4 Exclusive Owner

If a connection has a transport type of exclusive owner, it shall not be dependent on any other connection for its existence. For scheduled exclusive owner connections, the Forward_Open path shall contain a schedule segment. O⇒T data that controls outputs may be present; therefore, a target may only accept one exclusive owner connection. In addition, the target may accept listen only and input only connections that use the same multicast T⇒O data.

The term connection owner shall refer to the connection originator whose O⇒T packets are being consumed by the target object. The term owning connection shall refer to the connection associated with connection owner.

NOTE Exclusive owner connections are the most common transport type for controller to I/O connections since they allow the controller to receive inputs and control outputs on the same connection.

8.3.2 UCMM AR Formal Model

8.3.2.1 Description

The UCMM shall provide an unconnected request/response message service, limited to a single link that supports multiple outstanding messages. The required number of outstanding messages shall be implementation specific. The UCMM shall be present in all nodes, and shall support at least one outstanding message.

NOTE Most I/O modules need only support a single message while controllers typically support multiple outstanding messages. Setting up a connection requires communication between devices. This initial communication requires an unconnected message service. The UCMM, which only operates on a single link, is different from other transports. That is, while connected transport types use a single buffer to store packets, with new packets overwriting old ones, UCMM packets are queued at each node instead. Multiple connections can, therefore, be established at the same time. This reduces the overall time needed to set up all the connections in a system.

All incoming UCMM requests shall be internally forwarded to the Message Router, who is responsible for sending them to the target object instance or element, based on the path specified in the request.

8.3.2.2 Formal Class Definition

FAL ASE:	AR ASE
CLASS:	UCMM_Transaction_Record
CLASS ID:	<td>
PARENT CLASS:	TOP
ACCESS ATTRIBUTES:	
1	(m) Key Attribute: Record Number
STATIC ATTRIBUTES:	
1	(m) Attribute: Fixed Tag
2	(m) Attribute: Max_Retry
3	(m) Attribute: Retry Timeout
DYNAMIC ATTRIBUTES:	
1	(m) Attribute: Record State
2	(m) Attribute: Sequence Number
3	(m) Attribute: Retry Count
4	(m) Attribute: Response Timeout
SERVICES:	
1	(m) OpsService: UCMM_Create
2	(o) OpsService: UCMM_Delete
3	(m) OpsService: UCMM_Write
4	(m) OpsService: UCMM_Abort

8.3.2.2.1 Key Attributes

Record Number

Unique number associated with a given UCMM transaction record.

8.3.2.2.2 Static Attributes

These attributes have fixed values once the Transaction Record is created.

Fixed Tag

DLL fixed tag service used to exchange messages on this transaction record

Max_Retry

Maximum number of retries for this transaction record.

Retry Timeout

If no response or acknowledgement has been received back after the transmission of a UCMM packet (request or response) within the delay specified by the Retry Timeout, transmission of the packet should be repeated. The Retry Timeout value shall be fixed for the link at a value which guarantees that both the client and server nodes have an opportunity to transmit an unscheduled DLL packet (see IEC 61158-3).

8.3.2.2.3 Dynamic Attributes

These attributes change values based on the current operating state of the Transaction Record.

Record State

Current state of the Transaction Record. This shall be one of INACTIVE, RUNNING or WAITING-FOR-RESPONSE.

Sequence Number

Sequence Number shall be used for duplicate detection. It shall be incremented for every unique message on a given Transaction Record ; however it shall not be incremented on a retry.

The Sequence Number shall be set to zero at initialisation. The sequence number value shall be maintained in the INACTIVE state, to be used when the record transitions to running.

Retry Count

Current number of retries for a given message. When Retry count reaches Max_Retry, the current transaction shall be automatically aborted.

Response Timeout

The Response Timeout attribute indicates the maximum duration of the transaction in microseconds. If no response is received before the time-out expires, the transaction shall be automatically aborted. This value is provided when the UCMM_Write service is invoked.

8.3.2.2.4 Services

UCMM_Create

The UCMM_Create service shall attempt to create a UCMM transaction record.

UCMM_Delete

The UCMM_Delete service shall delete an existing transaction record.

UCMM_Write

The UCMM_Write service shall be used to send a message over an existing (already created) transaction record.

UCMM_Abort

The UCMM_Abort service shall be used either by the client or the server to abort a transaction.

8.3.3 Transport AR Formal Model

8.3.3.1 Description

The Transport AR's shall provide means of exchanging data along previously made connections. Depending on the functions performed by the connection, the Transport AR's fit into one of seven classes.

Applications interface to transport services through the supported transport "classes". Table 100 lists the general-purpose transport classes that have been defined for the systems. Each transport class provides a different level of functionality. Transport "class 0" provides the minimum functionality required of a transport class, enabling data transfer between applications.

Table 100 – Transport Classes

Class Number	Class Name
0	Null (or Base)
1	Duplicate Detection
2	Acknowledged
3	Verified
4	Non-blocking
5	Non-blocking, Fragmenting
6	Multicast, Fragmenting

8.3.3.2 Formal Class Definitions**8.3.3.2.1 Generic Transport AR**

This is the general model for all transport classes.

FAL ASE:**AR ASE****CLASS:**

Generic_Transport

CLASS ID:

<tbid>

PARENT CLASS:

TOP

ACCESS ATTRIBUTES:

- 1 (m) Key Attribute: Transport Identifier

STATIC ATTRIBUTES:

- 1 (m) Attribute: Transport class
 2 (o) Attribute: Max_Retry
 3 (o) Attribute: Retry Timeout

DYNAMIC ATTRIBUTES:

- 1 (m) Attribute: Transport State
 2 (o) Attribute: Sequence Count/Number
 3 (o) Attribute: Previous Sequence Number
 4 (o) Attribute: Command
 5 (o) Attribute: Previous Command
 6 (o) Attribute: Retry Count
 7 (o) Attribute: Response Timeout

SERVICES:

- 1 (o) OpService: TR_Write
 2 (o) OpService: TR_Trigger
 3 (o) OpService: TR_Packet_arrived
 4 (o) OpService: TR_Ack_received
 5 (o) OpService: TR_Verify
 6 (o) OpService: TR_Status_updated

8.3.3.2.1.1 Key Attribute**Transport identifier**

The Transport identifier is generated locally by the originator Connection Manager upon connection establishment, and serves as a further reference for the user to the established connection, i.e. this transport instance.

8.3.3.2.1.2 Static Attributes

These attributes have fixed values once the transport has been created as part of the connection establishment process. They are configured by the Connection Manager. These attributes are defined for management. It is not required for them to be visible to the communications system as accessible attributes.

Trans_class

The Trans_class attribute shall specify the transport class selected, and shall be one of NULL (also called "class 0" in both IEC 61158-5 and IEC 61158-6), DUPLICATE_DETECTION ("class 1"), ACKNOWLEDGED ("class 2"), VERIFIED ("class 3"), NON-BLOCKING ("class 4"), NON-BLOCKING_FRAGMENTING ("class 5"), or MULTICAST_FRAGMENTING ("class 6").

Max_Retry

Maximum number of retries for this transport instance.

Retry Timeout

If no response or acknowledgement has been received back after the transmission of a transport packet within the delay specified by the Retry Timeout, transmission of the packet should be repeated.

8.3.3.2.1.3 Dynamic Attributes

These attributes change values based on the current operating state of the transport instance.

Transport State

Current state of the transport instance. *This shall be one of NON-EXISTENT, IDLE, READY-TO-RUN, RUNNING, WAIT-FOR-ACKNOWLEDGE or WAIT-FOR-VERIFY.*

Sequence Count/Sequence Number

Sequence Count/Number shall be used for duplicate detection. It shall be incremented for every unique message on a given transport instance ; however it shall not be incremented on a retry.

The Sequence Count/Number shall be set to zero at initialisation.

Previous Sequence Number

This attribute shall save the Sequence Count/Number used by the previous message on this transport instance (used for duplicate detection).

Command

This attribute identifies the type of message sent on this transport instance, and shall be one of :

- NULL, ONLY, FIRST, MIDDLE, LAST for messages from the sender,
- ACK, NAK, NOT-INITIALISED for messages from the receiver.

Previous Command

This parameter shall save the Command value used by the previous message on this transport instance.

Retry Count

Current number of retries for a given message. When Retry count reaches Max_Retry, the current transaction shall be aborted.

Response Timeout

The Response Timeout attribute indicates the maximum duration of the transaction. If no response is received before the time-out expires, the transaction shall be aborted.

8.3.3.2.1.4 Services

TR_Write

The TR_Write service shall be used by a Client/Sender or Server application to put application data into a Transport PDU buffer (transport classes 0 to 3) or queue (transport classes 4 to 6) for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from a Client/Sender application. It shall be invoked when the data in the T-PDU buffer is to be sent.

TR_Packet_arrived

This service is invoked at a Server/Receiver application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer or queue.

TR_Ack_received

This service is invoked at a Client application to indicate that an acknowledgement has been received. New data (sent together with the acknowledgement) may be available in the T-PDU buffer.

TR_Verify

The TR_Verify service shall be used by a Server application to send back a verification response, after reception and processing of a request message from a Client application. This verification response may or may not contain data for the Client. This service is also invoked at a Client application to indicate that a verification has been received, and that new data (sent together with the verification) may be available in the T-PDU buffer.

TR_Status_updated

This service is invoked at a Sender application to indicate that status (success/error) for a previous request has been received, and is available in the dedicated Status queue.

NOTE As for all Type 2 objects, the services Create, Delete, Start, Stop and Reset are available for these transport AR's, but they are not exposed here since they are not directly accessible from the user application (only from the Connection Manager).

8.3.3.2.2 Class 0 Client Transport AR

Transport class 0 (Null transport type) is the simplest transport class, and can use either a point-to-point or multicast connection. Since transport class 0 does not detect duplicate data packets, the target application is responsible for detecting them.

Class 0 is typically used to transmit or receive inputs on a multicast connection or outputs on a point-to-point connection. Possible uses of transport class 0 include sampled inputs, standard outputs, cyclic block transfers, diagnostic events, and communication between controllers and operator interface devices.

This is the definition of the Class 0 Client Transport (Null transport type).

FAL ASE: **AR ASE**
CLASS: Transport_Client_0
CLASS ID: <tbid>
PARENT CLASS: Generic_Transport

ACCESS ATTRIBUTES:

1 (m) Key Attribute: Transport identifier

STATIC ATTRIBUTES:

1 (m) Attribute: Transport class = NULL

DYNAMIC ATTRIBUTES:

1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | RUNNING

SERVICES:

1 (m) OpService: TR_Write
 2 (m) OpService: TR_Trigger

8.3.3.2.2.1 Attributes

No specific requirement for this object.

8.3.3.2.2.2 Services

TR_Write

The TR_Write service shall be used by the Client application to put application data into a Transport PDU buffer for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from the Client application. It shall be invoked when the data in the T-PDU buffer is to be sent.

8.3.3.2.3 Class 0 Server Transport AR

This is the definition of the Class 0 Server Transport (Null transport type).

FAL ASE: **AR ASE**
CLASS: Transport_Server_0
CLASS ID: <tbid>
PARENT CLASS: Generic_Transport

ACCESS ATTRIBUTES:

1 (m) Key Attribute: Transport identifier

STATIC ATTRIBUTES:

1 (m) Attribute: Transport class

DYNAMIC ATTRIBUTES:

1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | RUNNING

SERVICES:

3 (m) OpService: TR_Packet_arrived

8.3.3.2.3.1 Attributes

No specific requirement for this object.

8.3.3.2.3.2 Services

TR_Packet_arrived

This service is invoked at the Server application to indicate that a new packet has been received, but it will not detect duplicate packets in this case. The new packet is available in the T-PDU buffer.

8.3.3.2.4 Class 1 Client Transport AR

Transport class 1 (Duplicate Detection transport type), like class 0, shall allow either a point-to-point or multicast connection. Unlike class 0, the class 1 transport allows to detect the delivery of duplicate data packets.

NOTE Class 1 is preferred to class 0 for targets because the target application does not have to perform duplicate detection, which reduces the overhead required in those end nodes that support the change of state transport trigger. Class 1 permits an efficient change of state trigger, since was designed to allow change of state data transfers.

Applications could use this transport class to distinguish between new samples and old samples that were sent to maintain the connection. To maintain the connection, a timer may be used to trigger the production of the current data in the T-PDU buffer. This may allow applications to reduce their overhead by only processing new data samples.

Possible uses of transport class 1 include sampled inputs, standard outputs, cyclic block transfers, diagnostic events, and communication between controllers and operator interface devices.

This is the definition of the Class 1 Client Transport (Duplicate Detection transport type).

FAL ASE:

AR ASE

CLASS:

Transport_Client_1

CLASS ID:

<tbid>

PARENT CLASS:

Generic_Transport

ACCESS ATTRIBUTES:

1 (m) Key Attribute: Transport identifier

STATIC ATTRIBUTES:

1 (m) Attribute: Transport class = DUPLICATE_DETECTION

DYNAMIC ATTRIBUTES:

1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | RUNNING

2 (m) Attribute: Sequence Count/Number

SERVICES:

1 (m) OpsService: TR_Write

2 (m) OpsService: TR_Trigger

8.3.3.2.4.1 Attributes

No specific requirement for this object.

8.3.3.2.4.2 Services

TR_Write

The TR_Write service shall be used by the Client application to put application data into a Transport PDU buffer for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from the Client application. It shall be invoked when the data in the T-PDU buffer is to be sent.

8.3.3.2.5 Class 1 Server Transport AR

This is the definition of the Class 1 Server Transport (Duplicate Detection transport type).

FAL ASE:	AR ASE
CLASS:	Transport_Server_1
CLASS ID:	<tbd>
PARENT CLASS:	Generic_Transport
ACCESS ATTRIBUTES:	
1 (m) Key Attribute:	Transport identifier
STATIC ATTRIBUTES:	
1 (m) Attribute:	Transport class = DUPLICATE_DETECTION
DYNAMIC ATTRIBUTES:	
1 (m) Attribute:	Transport State = NON-EXISTENT IDLE READY-TO-RUN RUNNING
2 (m) Attribute:	Sequence Count/Number
3 (m) Attribute:	Previous Sequence Number
SERVICES:	
3 (m) OpsService:	TR_Packet_arrived

8.3.3.2.5.1 Attributes

No specific requirement for this object.

8.3.3.2.5.2 Services

TR_Packet_arrived

This service is invoked at the Server application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer.

8.3.3.2.6 Class 2 Client Transport AR

Transport class 2 (Acknowledged transport type) starts with the behaviour in class 1 and adds a return connection that acknowledges the delivery of a packet. The producer connection from the client node can be point to point or multicast. The connection from the server to the client is point to point.

This transport class allows the application to be notified that the server has received the transmitted packet. This acknowledgement tells the client application only that the data arrived. This does not mean that the server application has read the buffer or that a new sample can be sent without overwriting the buffer.

The main advantage of this class over class 3 (Verified) is that the acknowledgement is returned immediately after the packet arrives at the consumer, while the verify class requires the application to initiate the return verification. The delays associated with receiving the acknowledgement are small, approximately twice the time that is required for a one way transmission.

Transport class 2 uses two connections: the originator-to-target connection, which can be either point-to-point or multicast; and the return connection, which shall be point-to-point. A multicast connection using transport class 2 shall have a point-to-point return connection corresponding to

each client-to-server connection. The acknowledgement notifies the client transport instance that the consuming node of the network connection has received the packet.

If a server transport instance does not acknowledge receipt of the packet, the client transport instance triggers the client application to retransmit the most recently sent packet; if the client-to-server connection is one of the network connections of a multicast, data is resent over all of the network connections of the multicast. The server application can return data to the client application along with its acknowledgement.

Possible uses of transport class 2 include master/slave applications and communication between controllers and operator interface devices.

This is the definition of the Class 2 Client Transport (Acknowledged transport type).

FAL ASE:	AR ASE
CLASS:	Transport_Client_2
CLASS ID:	<tbid>
PARENT CLASS:	Generic_Transport
ACCESS ATTRIBUTES:	
1 (m) Key Attribute:	Transport identifier
STATIC ATTRIBUTES:	
1 (m) Attribute:	Transport class = ACKNOWLEDGED
2 (m) Attribute:	Max_Retry
3 (m) Attribute:	Retry Timeout
DYNAMIC ATTRIBUTES:	
1 (m) Attribute:	Transport State = NON-EXISTENT IDLE RUNNING WAIT-FOR-ACKNOWLEDGE
2 (m) Attribute:	Sequence Count/Number
6 (m) Attribute:	Retry Count
7 (m) Attribute:	Response Timeout
SERVICES:	
1 (m) OpsService:	TR_Write
2 (m) OpsService:	TR_Trigger
4 (m) OpsService:	TR_Ack_received

8.3.3.2.6.1 Attributes

No specific requirement for this object.

8.3.3.2.6.2 Services

TR_Write

The TR_Write service shall be used by the Client application to put application data into a Transport PDU buffer for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from the Client application. It shall be invoked when the data in the T-PDU buffer is to be sent.

TR_Ack_received

This service is invoked at the Client application to indicate that an acknowledgement has been received. New data (sent together with the acknowledgement) may be available in the T-PDU buffer.

8.3.3.2.7 Class 2 Server Transport AR

This is the definition of the Class 2 Server Transport (Acknowledged transport type).

FAL ASE:	AR ASE
CLASS:	Transport_Server_2
CLASS ID:	<tbid>
PARENT CLASS:	Generic_Transport
ACCESS ATTRIBUTES:	
1 (m) Key Attribute:	Transport identifier
STATIC ATTRIBUTES:	
1 (m) Attribute:	Transport class = ACKNOWLEDGED
DYNAMIC ATTRIBUTES:	
1 (m) Attribute:	Transport State = NON-EXISTENT IDLE READY-TO-RUN RUNNING
2 (m) Attribute:	Sequence Count/Number
3 (m) Attribute:	Previous Sequence Number
SERVICES:	
1 (m) OpService:	TR_Write
3 (m) OpService:	TR_Packet_arrived

8.3.3.2.7.1 Attributes

No specific requirement for this object.

8.3.3.2.7.2 Services

TR_Write

The TR_Write service shall be used by the Server application to put application data into a Transport PDU buffer for further transmission together with the acknowledgement.

TR_Packet_arrived

This service is invoked at the Server application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer.

8.3.3.2.8 Class 3 Client Transport AR

Transport class 3 (Verified transport type) starts with the behaviour in class 1 and adds a return connection that verifies that the application has received the packet. The sequence count that shall be returned with the verify is the same as the sequence count sent with the data. This return connection is used to identify the verification message.

This transport class allows the application to be notified that the server application has responded to the transmitted packet. This verification tells the client application not only that the data arrived but that the server application has read the buffer and that a new sample can be sent without overwriting the buffer.

The main advantage of this class over class 2 (Acknowledgement) is that verification conveys an application response. The disadvantage is that the delay in receiving the verification can be very difficult to predict.

Transport class 3 uses two connections: the originator-to-target connection, which can be either point-to-point or multicast; and the return connection, which shall be point-to-point. A multicast connection using transport class 3 shall have a point-to-point return connection corresponding to each client-to-server connection. The verification notifies the client application that the server application has received and read the transmitted data. If a server application does not verify receipt of the packet, the client application retransmits the most recently sent packet; if the client-to-server connection is a multicast, the data is retransmitted over all of the network connections of the multicast.

Possible uses of transport class 3 include change-of-state inputs and outputs, asynchronous [independent] block transfers, event acknowledgements, communication between controllers and operator interface devices, uploads, downloads, and messaging.

This is the definition of the Class 3 Client Transport (Verified transport type).

FAL ASE:	AR ASE
CLASS:	Transport_Client_3
CLASS ID:	<tbid>
PARENT CLASS:	Generic_Transport
ACCESS ATTRIBUTES:	
1 (m) Key Attribute:	Transport identifier
STATIC ATTRIBUTES:	
1 (m) Attribute:	Transport class = VERIFIED
2 (m) Attribute:	Max_Retry
3 (m) Attribute:	Retry Timeout
DYNAMIC ATTRIBUTES:	
1 (m) Attribute:	Transport State = NON-EXISTENT IDLE RUNNING WAIT-FOR-VERIFY
2 (m) Attribute:	Sequence Count/Number
6 (m) Attribute:	Retry Count
7 (m) Attribute:	Response Timeout
SERVICES:	
1 (m) OpService:	TR_Write
2 (m) OpService:	TR_Trigger
5 (m) OpService:	TR_Verify

8.3.3.2.8.1 Attributes

No specific requirement for this object.

8.3.3.2.8.2 Services

TR_Write

The TR_Write service shall be used by the Client application to put application data into a Transport PDU buffer for further transmission.

TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from the Client application. It shall be invoked when the data in the T-PDU buffer is to be sent.

TR_Verify

This service is invoked at the Client application to indicate that a verification has been received, and that new data (sent together with the verification) may be available in the T-PDU buffer.

8.3.3.2.9 Class 3 Server Transport AR

This is the definition of the Class 3 Server Transport (Verified transport type).

FAL ASE:	AR ASE
CLASS:	Transport_Server_3
CLASS ID:	<tbd>
PARENT CLASS:	Generic_Transport
ACCESS ATTRIBUTES:	
1 (m) Key Attribute:	Transport identifier
STATIC ATTRIBUTES:	
1 (m) Attribute:	Transport class = VERIFIED
DYNAMIC ATTRIBUTES:	
1 (m) Attribute:	Transport State = NON-EXISTENT IDLE READY-TO-RUN WAIT-FOR-VERIFY RUNNING
2 (m) Attribute:	Sequence Count/Number
3 (m) Attribute:	Previous Sequence Number
SERVICES:	
1 (m) OpService:	TR_Write
3 (m) OpService:	TR_Packet_arrived
5 (m) OpService:	TR_Verify

8.3.3.2.9.1 Attributes

No specific requirement for this object.

8.3.3.2.9.2 Services

TR_Write

The TR_Write service shall be used by the Server application to put application data into a Transport PDU buffer for further transmission together with the verification.

TR_Packet_arrived

This service is invoked at the Server application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer.

TR_Verify

The TR_Verify service shall be used by the Server application to send back a verification response, after reception and processing of a request message from the Client application. This verification response may or may not contain data for the Client.

8.3.3.2.10 Class 4 Sender Transport AR

Transport class 4 (Non-blocking transport type) is similar to having two class 2 transports, one in each direction, except that the class 4 uses two Network Connections instead of four. Transport class 4 can transfer application information in both directions. The delivery is acknowledged by the receiving transport. It uses a sequence number to identify which T-PDU is being acknowledged. Retries are performed.

The main advantage of this class over using two Class 2 (Acknowledgement) transports is that class 4 uses fewer resources. The main advantage of class 4 over class 3 is that it does not block the application from sending a second request before the first response is received. This pipelining can improve the system performance in some circumstances. In addition, class 4 supports bi-directional, asynchronous data transmission, without imposing request-response type serialisation.

This transport class notifies the application that the server application has responded to the transmitted packet. This transport class shall automatically send the next item in the queue when it receives an acknowledge that the previous one has been acknowledged. The application shall be notified if the network connection fails.

Transport class 4 uses two connections: the originator-to-target connection, which shall be point-to-point; and the return connection, which also shall be point-to-point.

The receiving transport initiates and sends the acknowledgement after the consuming node of the network connection has written the packet to the T-PDU buffer, after the transport has read the packet's transport header, and after the transport has placed a copy of the data into a queue for the application. The receiver's acknowledgement signals the sending transport that the receiver-side T-PDU receive buffer is ready to accept the next data packet.

The limit to the number of application requests which can be sent before an application response is received is determined by the design of the target application. It shall typically be limited by the amount of memory which the receiving transport can allocate for holding the outstanding messages. An error shall be returned to the sending transport when a transmission attempts to exceed the limit.

Transport class 4 shall not be used with messaging. Messaging does not provide a way to match requests with responses. Use class 3 for messaging without fragmentation.

This is the definition of the Class 4 Sender Transport (Non-blocking transport type).

FAL ASE:

AR ASE

CLASS:

Transport_Sender_4

CLASS ID:

<tbd>

PARENT CLASS:

Generic_Transport

ACCESS ATTRIBUTES:

- 1 (m) Key Attribute: Transport identifier

STATIC ATTRIBUTES:

- 1 (m) Attribute: Transport class = NON-BLOCKING
 2 (m) Attribute: Max_Retry
 3 (m) Attribute: Retry Timeout

DYNAMIC ATTRIBUTES:

- 1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | READY | WAITING
 2 (m) Attribute: Sequence Count/Number
 4 (m) Attribute: Command
 6 (m) Attribute: Retry Count
 7 (m) Attribute: Response Timeout

SERVICES:

- 1 (m) OpsService: TR_Write
 6 (m) OpsService: TR_Status_updated

NOTE The TR_Trigger service is not listed here since it is only triggered internally by the transport instance timer, in order to send a retry or a "keep alive" message.

8.3.3.2.10.1 Attributes

No specific requirement for this object.

8.3.3.2.10.2 Services

TR_Write

The TR_Write service shall be used by the Sender application to put application data into a Transport PDU queue for further transmission.

TR_Status_updated

This service is invoked at a Sender application to indicate that status (success/error) for a previous request has been received, and is available in the dedicated Status queue.

8.3.3.2.11 Class 4 Receiver Transport AR

This is the definition of the Class 4 Receiver Transport (Non-blocking transport type).

FAL ASE:	AR ASE
CLASS:	Transport_Receiver_4
CLASS ID:	<tbd>
PARENT CLASS:	Generic_Transport
ACCESS ATTRIBUTES:	
1 (m) Key Attribute:	Transport identifier
STATIC ATTRIBUTES:	
1 (m) Attribute:	Transport class = NON-BLOCKING
DYNAMIC ATTRIBUTES:	
1 (m) Attribute:	Transport State = NON-EXISTENT IDLE INITIAL READY
2 (m) Attribute:	Sequence Count/Number
3 (m) Attribute:	Previous Sequence Number
4 (m) Attribute:	Command
5 (m) Attribute:	Previous Command
SERVICES:	
3 (m) OpsService:	TR_Packet_arrived

8.3.3.2.11.1 Attributes

No specific requirement for this object.

8.3.3.2.11.2 Services

TR_Packet_arrived

This service is invoked at the Receiver application to indicate that a new packet has been received. The new packet is available in the T-PDU queue.

8.3.3.2.12 Class 5 Sender Transport AR

Transport class 5 (Non-blocking, Fragmenting transport type) is similar to class 4, except that class 5 adds Fragmentation and Reassembly at the transport layer. Transport class 5 can transfer application information in both directions. The delivery of each fragment is acknowledged by the receiving transport. A sequence number identifies which T-PDU is being acknowledged. Retries are performed.

This transport class notifies the application that the server application has responded to the transmitted packet. This transport class shall automatically send the next item in the queue when it receives an acknowledgement that the previous one has been acknowledged. The application shall be notified if the network connection fails.

Transport Class 5 uses two connections: the originator-to-target connection, which shall be point-to-point, and the return connection, which also shall be point-to-point.

The receiving transport initiates and sends the acknowledgement after the consuming node of the network connection has written the packet to the T-PDU buffer, after the transport has read the packet's transport header, and after the transport has placed a copy of the data into a queue for the application. The receiver's acknowledgement signals the sending transport that the receiver-side T-PDU receive buffer is ready to accept the next data packet.

The limit to the number of application requests which can be sent before an application response is received is determined by the design of the target application. It shall typically be limited by the amount of memory. An error shall be returned when a transmission attempts to exceed the limit.

Transport class 5 shall not be used with messaging. Messaging does not provide a way to match requests with responses. Use class 3 for messaging without fragmentation.

The fragmentation protocols assume that per-connection send order is preserved throughout the system; overwrite is permissible, but changing the order of the messages is not.

Possible uses of transport class 5 include exchanging messages and responses up to 65536 bytes between two applications.

This is the definition of the Class 5 Sender Transport (Non-blocking, Fragmenting transport type).

FAL ASE:

AR ASE

CLASS:

Transport_Sender_5

CLASS ID:

<tbid>

PARENT CLASS:

Generic_Transport

ACCESS ATTRIBUTES:

- 1 (m) Key Attribute: Transport identifier

STATIC ATTRIBUTES:

- 1 (m) Attribute: Transport class = NON-BLOCKING_FRAGMENTING
- 2 (m) Attribute: Max_Retry
- 3 (m) Attribute: Retry Timeout

DYNAMIC ATTRIBUTES:

- 1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | READY | WAITING
- 2 (m) Attribute: Sequence Count/Number
- 4 (m) Attribute: Command
- 6 (m) Attribute: Retry Count
- 7 (m) Attribute: Response Timeout

SERVICES:

- 1 (m) OpsService: TR_Write
- 6 (m) OpsService: TR_Status_updated

NOTE The TR_Trigger service is not listed here since it is only triggered internally by the transport instance timer, in order to send a retry or a "keep alive" message.

8.3.3.2.12.1 Attributes

No specific requirement for this object.

8.3.3.2.12.2 Services

TR_Write

The TR_Write service shall be used by the Sender application to put application data into a Transport PDU queue for further transmission.

TR_Status_updated

This service is invoked at a Sender application to indicate that status (success/error) for a previous request has been received, and is available in the dedicated Status queue.

8.3.3.2.13 Class 5 Receiver Transport AR

This is the definition of the Class 5 Receiver Transport (Non-blocking, Fragmenting transport type).

FAL ASE:	AR ASE
CLASS:	Transport_Receiver_5
CLASS ID:	<tbd>
PARENT CLASS:	Generic_Transport
ACCESS ATTRIBUTES:	
1 (m) Key Attribute:	Transport identifier
STATIC ATTRIBUTES:	
1 (m) Attribute:	Transport class = NON-BLOCKING_FRAGMENTING
DYNAMIC ATTRIBUTES:	
1 (m) Attribute:	Transport State = NON-EXISTENT IDLE INITIAL ASSEMBLING READY
2 (m) Attribute:	Sequence Count/Number
3 (m) Attribute:	Previous Sequence Number
4 (m) Attribute:	Command
5 (m) Attribute:	Previous Command
SERVICES:	
3 (m) OpsService:	TR_Packet_arrived

8.3.3.2.13.1 Attributes

No specific requirement for this object.

8.3.3.2.13.2 Services

TR_Packet_arrived

This service is invoked at the Receiver application to indicate that a new packet has been received. The new packet is available in the T-PDU queue.

8.3.3.2.14 Class 6 Client Transport AR

Transport class 6 (Multicast, Fragmenting transport type) is similar to class 3. Transport class 6 can transfer application information in both directions. Class 6 supports multicasting in the Client to Server direction. Class 6 adds Fragmentation and Reassembly at the transport layer, but only in the Client to Server (multicast) direction.

The delivery of each fragment is acknowledged by the receiving transport. A sequence number identifies the T-PDU being acknowledged. Retries are performed.

Class 6 is NOT a superset of class 5. Class 6 is blocking; that is, the next application request is not delivered until the current one is acknowledged. Functionally, class 6 shall be thought of as a fragmenting version of class 3, rather than as a multicasting version of class 5. However, class 6 reuses the transport header format from class 5, instead of the class 3 transport header.

This transport class notifies the application that the server application has responded to the transmitted packet. The application shall be notified if the network connection fails.

Transport class 6 uses multiple network connections: the Client-to-Server connection, which may be multicast; and the return connections (one per Server), which shall be point-to-point.

One significant difference between class 3 and class 6, which results from class 6 doing fragmentation: the "write" event causes the first (or only) fragment to be transmitted; the "trigger" event is only a timer function which is used to signal retries or keep-alive transmissions.

The fragmentation protocols assume that per-connection send order is preserved throughout the system; overwrite is permissible, but changing the order of the messages is not.

Possible uses of transport class 6 include delivering event notification messages (up to 65536 bytes) to multiple servers.

This is the definition of the Class 6 Client Transport (Multicast, Fragmenting transport type).

FAL ASE:

AR ASE

CLASS:

Transport_Client_6

CLASS ID:

<td>

PARENT CLASS:

Generic_Transport

ACCESS ATTRIBUTES:

1 (m) Key Attribute: Transport identifier

STATIC ATTRIBUTES:

1 (m) Attribute: Transport class = MULTICAST_FRAGMENTING

2 (m) Attribute: Max_Retry

3 (m) Attribute: Retry Timeout

DYNAMIC ATTRIBUTES:

1 (m) Attribute: Transport State = NON-EXISTENT | IDLE | READY | WAITING

2 (m) Attribute: Sequence Count/Number

4 (m) Attribute: Command

6 (m) Attribute: Retry Count

7 (m) Attribute: Response Timeout

SERVICES:

1 (m) OpsService: TR_Write

5 (m) OpsService: TR_Verify

NOTE The TR_Trigger service is not listed here since it is only triggered internally by the transport instance timer, in order to send a retry or a "keep alive" message.

8.3.3.2.14.1 Attributes

No specific requirement for this object.

8.3.3.2.14.2 Services

TR_Write

The TR_Write service shall be used by the Client application to put application data into a Transport PDU queue for further transmission.

TR_Verify

This service is invoked at the Client application to indicate that a verification has been received, and that new data (sent together with the verification) may be available in the T-PDU queue.

8.3.3.2.15 Class 6 Server Transport AR

This is the definition of the Class 6 Server Transport (Multicast, Fragmenting transport type).

FAL ASE:	AR ASE
CLASS:	Transport_Server_6
CLASS ID:	<tbid>
PARENT CLASS:	Generic_Transport
ACCESS ATTRIBUTES:	
1 (m) Key Attribute:	Transport identifier
STATIC ATTRIBUTES:	
1 (m) Attribute:	Transport class = MULTICAST_FRAGMENTING
2 (m) Attribute:	Max_Retry
3 (m) Attribute:	Retry Timeout
DYNAMIC ATTRIBUTES:	
1 (m) Attribute:	Transport State = NON-EXISTENT IDLE INITIAL ASSEMBLING WAITING READY
2 (m) Attribute:	Sequence Count/Number
3 (m) Attribute:	Previous Sequence Number
4 (m) Attribute:	Command
5 (m) Attribute:	Previous Command
6 (m) Attribute:	Retry Count
7 (m) Attribute:	Response Timeout
SERVICES:	
1 (m) OpsService:	TR_Write
3 (m) OpsService:	TR_Packet_arrived
5 (m) OpsService:	TR_Verify

8.3.3.2.15.1 Attributes

No specific requirement for this object.

8.3.3.2.15.2 Services

TR_Write

The TR_Write service shall be used by the Server application to put application data into a Transport PDU queue for further transmission together with the verification.

TR_Packet_arrived

This service is invoked at the Server application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU queue.

TR_Verify

The TR_Verify service shall be used by the Server application to send back a verification response, after reception and processing of a request message from the Client application. This verification response may or may not contain data for the Client.

8.3.4 AR ASE Services

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

- UCMM_Create
- UCMM_Delete
- UCMM_Write
- UCMM_Abort
- TR_Write
- TR_Trigger
- TR_Packet_arrived
- TR_Ack_received
- TR_Verify
- TR_Status_updated

8.3.4.1.1 Common parameters**Record Number**

This parameter is the unique number associated with the UCMM transaction record to be used for the current transaction.

Transport identifier

The Transport identifier is generated locally by the originator Connection Manager upon connection establishment, and serves as a further reference for the user to the established connection, i.e. this transport instance.

8.3.4.1.2 UCMM_Create

The UCMM_Create service request shall attempt to create a UCMM transaction record.

8.3.4.1.2.1 Service Primitives

The service parameters for this service are shown in Table 101.

Table 101 – UCMM_Create Service Parameters

Parameter name	Req	Cnf
Argument	M	
Fixed Tag	M	
Max_Retry	M	
Result		M
Record Number		M
Service Status		M

Argument

The argument contains the parameters of the service request.

Fixed Tag

DLL fixed tag service used to exchange messages on this transaction record (regular UCMM or management UCMM).

Max_Retry

Maximum number of retries for this transaction record.

Result

This selection type parameter indicates whether the service request succeeded or failed.

Record Number

This parameter is the unique number returned by the UCMM and associated with the newly created UCMM transaction record.

Service Status

This parameter provides information on the result of service execution. It shall be one of SUCCESS or CANNOT_CREATE.

8.3.4.1.2.2 Service Procedure

The UCMM_Create service request from the user is processed locally by the UCMM. This service may also be invoked by internal objects (e.g. the Connection Manager).

A transaction record must be created before any message can be exchanged through the UCMM. Only one message shall be outstanding for a given transaction record. If multiple outstanding messages are required by the user application, then multiple transaction records are required.

8.3.4.1.3 UCMM_Delete

The UCMM_Delete service shall delete an existing transaction record.

8.3.4.1.3.1 Service Primitives

The service parameters for this service are shown in Table 102.

Table 102 – UCMM_Delete Service Parameters

Parameter name	Req
Argument	M
Record Number	M

Argument

The argument contains the parameters of the service request.

Record Number

This parameter is the unique number associated with the UCMM transaction record to be deleted.

8.3.4.1.3.2 Service Procedure

The UCMM_Delete service request from the user is processed locally by the UCMM. This service may also be invoked by internal objects (e.g. the Connection Manager).

8.3.4.1.4 UCMM_Write

The UCMM_Write service shall be used to send a message over an existing (already created) transaction record.

8.3.4.1.4.1 Service Primitives

The service parameters for this service are shown in Table 103.

Table 103 – UCMM_Write Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M		
Record Number	M	M(=)		
Dest_ID	M			
Source_ID		M		
UCMM_service	M			
Response Timeout	M			
Transaction_Priority	M			
App_Data	M	M(=)		
Result			M	M
Record Number			M	M(=)
Service Status				M
App_Data			M	M(=)

Argument

The argument contains the parameters of the service request.

Dest_ID

This is the MAC ID of the target (server) device for this transaction.

Source_ID

This is the MAC ID of the originating (client) device for this transaction.

UCMM_Service

The UCMM_Service parameter shall specify the delivery mechanism to use for this message and shall be one of :

- Request_With_Acknowledge (request with retry until acknowledged)
- Request_With_No_Response (request with no acknowledgement and no response)
- Request_With_No_ACK (request with retry until response – no acknowledgement)
- Request_With_No_Retry (request with no retry, and response with no retry)

NOTE These values correspond to request command codes contained in the UCMM header (see IEC 61158-6).

Response Timeout

The Response Timeout attribute indicates the maximum duration of the transaction in microseconds. If no response is received before the time-out expires, the transaction shall be automatically aborted, and the Service Status parameter of the confirmation shall be TIMEOUT.

Transaction_Priority

The Transaction_priority parameter shall determine which Data Link Layer priority to use for the messages, and shall be one of LOW or HIGH.

NOTE The definition of DLL priority is specified in IEC 61158-4.

App_Data

The App_Data parameter contains application data (service request parameters). The corresponding data length shall be less than 504 bytes since any UCMM packet shall fit into a single Lpacket.

Result

This selection type parameter indicates whether the service request succeeded or failed.

Service Status

This parameter provides information on the result of service execution. It shall be one of :

SUCCESS
 RECORD_NOT_CREATED
 PACKET_TOO_BIG
 NO_ACKNOWLEDGE
 ABORTED
 TIMEOUT

App_Data

The App_Data parameter contains application data (service response parameters). The corresponding data length shall be less than 504 bytes since any UCMM packet shall fit into a single Lpacket.

8.3.4.1.4.2 Service Procedure

The UCMM_Write request shall send a message to the device on the local link specified by the Dest_ID parameter. When the message arrives at the target device, this shall result in a UCMM_Write indication at the Message Router in that device (the Message Router shall receive all UCMM_Write indications). The Message Router, in turn, shall forward the message to the appropriate target object. When the object responds, the Message Router shall forward the response via the UCMM_Write response primitive. When the response arrives at the device which originated the request, a UCMM_Write confirmation shall be returned to the application object which sent the request.

8.3.4.1.5 UCMM_Abort

The UCMM_Abort service shall be used either by the client or the server to abort a transaction.

8.3.4.1.5.1 Service Primitives

The service parameters for this service are shown in Table 104.

Table 104 – UCMM_Abort Service Parameters

Parameter name	Req
Argument	M
Record Number	M

Argument

The argument contains the parameters of the service request.

Record Number

This parameter is the unique number associated with the UCMM transaction record for which the pending transaction is to be aborted.

8.3.4.1.5.2 Service Procedure

The UCMM_Abort service request from the user is processed locally by the UCMM. This service may also be invoked by internal objects.

When called by the client, the UCMM_Abort service shall abort the transaction corresponding to the specified transaction record, which shall remove the packet from the local DLL queue if it has not yet been transmitted. When called by the server, this request shall abort the transaction, which shall suppress the response from being sent. This service need not have a corresponding confirmation since the confirmation of the original request shall serve this purpose by returning a status of ABORTED.

8.3.4.1.6 TR_Write

The TR_Write service shall be used by a Client/Sender or Server application to put application data into a Transport PDU buffer (transport classes 0 to 3) or queue (transport classes 4 to 6) for further transmission. The data written into the Transport PDU buffer or queue has no fixed specification or format.

NOTE Usage of PDU buffers and queues is described in IEC 61158-6.

8.3.4.1.6.1 Service Primitives

The service parameters for this service are shown in Table 105.

Table 105 – TR_Write Service Parameters

Parameter name	Req
Argument	M
Transport identifier	M
App_Data	M

Argument

The argument contains the parameters of the service request.

App_Data

The App_Data parameter contains application data (service request/response parameters or user formatted data).

8.3.4.1.6.2 Service Procedure

Since they are dependant on the transport class selected, service procedures for this service will be detailed in IEC 61158-6.

8.3.4.1.7 TR_Trigger

The TR_Trigger service initiates the actual transfer of Transport PDU from a Client/Sender application. It shall be invoked when the data in the T-PDU buffer is to be sent. Depending on the type of trigger selected at connection establishment for this transport instance, this service can be requested either directly by the user application, or internally by a timer associated with the connection.

8.3.4.1.7.1 Service Primitives

The service parameters for this service are shown in Table 106.

Table 106 – TR_Trigger Service Parameters

Parameter name	Req
Argument	M
Transport identifier	M

Argument

The argument contains the parameters of the service request.

8.3.4.1.7.2 Service Procedure

Since they are dependant on the transport class selected, service procedures for this service will be detailed in IEC 61158-6.

8.3.4.1.8 TR_Packet_arrived

This service is invoked at a Server/Receiver application to indicate that a new packet has been received, and whether it is a duplicate or not. The new packet is available in the T-PDU buffer or queue.

8.3.4.1.8.1 Service Primitives

The service parameters for this service are shown in Table 107.

Table 107 – TR_Packet_arrived Service Parameters

Parameter name	Ind
Argument	M
Transport identifier	M
Data_Arrived	M
Duplicate_Arrived	M

Argument

The argument contains the parameters of the service indication.

Data_Arrived / Duplicate_Arrived

The Data_Arrived and Duplicate_Arrived parameters indicate whether the new packet contains new data, or whether it is just a duplicate from a previously received packet.

NOTE A Duplicate Arrived event is normally not of interest to the application; however, it might be useful in triggering a watchdog timer to indicate that the client's application is alive and triggering.

8.3.4.1.8.2 Service Procedure

Since they are dependant on the transport class selected, service procedures for this service will be detailed in IEC 61158-6.

8.3.4.1.9 TR_Ack_received

This service is invoked at a Client application to indicate that an acknowledgement has been received. New data (sent together with the acknowledgement) may be available in the T-PDU buffer.

8.3.4.1.9.1 Service Primitives

The service parameters for this service are shown in Table 108.

Table 108 – TR_Ack_received Service Parameters

Parameter name	Cnf
Result	M
Transport identifier	M

Result

This parameter provides information for the receiving application.

8.3.4.1.9.2 Service Procedure

Since they are dependant on the transport class selected, service procedures for this service will be detailed in IEC 61158-6.

8.3.4.1.10 TR_Verify

The TR_Verify service shall be used by a Server application to send back a verification response, after reception and processing of a request message from a Client application. This verification response may or may not contain data for the Client. This service is also invoked at a Client application to indicate that a verification has been received, and that new data (sent together with the verification) may be available in the T-PDU buffer.

8.3.4.1.10.1 Service Primitives

The service parameters for this service are shown in Table 109.

Table 109 – TR_Verify Service Parameters

Parameter name	Rsp	Cnf
Result	M	M
Transport identifier	M	M

Result

This parameter provides information for the receiving application.

8.3.4.1.10.2 Service Procedure

Since they are dependant on the transport class selected, service procedures for this service will be detailed in IEC 61158-6.

8.3.4.1.11 TR_Status_updated

This service is invoked at a Sender application to indicate that status (success/error) for a previous request has been received, and is available in the dedicated Status queue.

8.3.4.1.11.1 Service Primitives

The service parameters for this service are shown in Table 110.

Table 110 – TR_Status_updated Service Parameters

Parameter name	Cnf
Result	M
Transport identifier	M

Result

This parameter provides information for the receiving application.

8.3.4.1.11.2 Service Procedure

Since they are dependant on the transport class selected, service procedures for this service will be detailed in IEC 61158-6.

8.4 Summary of FAL Type 2 Classes

This annex contains a summary of the defined FAL Classes. The Class ID values have been assigned to be compatible with existing standards.

Table 111– FAL Class Summary

FAL ASE	Class	Class ID
OM ASE	General Type 2	tbd
	Identity	tbd
	Assembly	tbd
	Message Router	tbd
CM ASE	Connection Manager	tbd
AR ASE	UCMM – Client	tbd
	UCMM – Server	tbd
	Generic Transport	tbd
	Transport Class 0 – Client	tbd
	Transport Class 0 – Server	tbd
	Transport Class 1 – Client	tbd
	Transport Class 1 – Server	tbd
	Transport Class 2 – Client	tbd
	Transport Class 2 – Server	tbd
	Transport Class 3 - Client	tbd
	Transport Class 3 – Server	tbd
	Transport Class 4 - Sender	tbd
	Transport Class 4 – Receiver	tbd
	Transport Class 5 – Sender	tbd
	Transport Class 5 – Recevier	tbd
	Transport Class 6 - Client	tbd
	Transport Class 6 - Server	tbd

8.5 Permitted FAL Type 2 Services by AR Type

Table 112 below defines the valid combinations of services and AR types. EP roles (which service APDUs and AR with the specified type can send or receive). The Unc and Cnf columns indicate whether the service listed in the left-hand column is unconfirmed (Unc) or confirmed (Cnf).

Table 112 – FAL Type 2 Services by AR Type

FAL Services	Unc	Cnf	Client / Sender		Server / Receiver		UCMM	TR0 - TR6
			req	rcv	req	rcv		
OM ASE								
Get_Attribute_Single		X	X		X		X	TR3
Get_Attributes_List		X	X		X		X	TR3
Get_Attributes_All		X	X		X		X	TR3
Set_Attribute		X	X		X		X	TR3
Set_Attributes_List		X	X		X		X	TR3
Set_Attributes_All		X	X		X		X	TR3
Create		X	X		X		X	TR3
Delete		X	X		X		X	TR3
Start		X	X		X		X	TR3
Stop		X	X		X		X	TR3
Reset		X	X		X		X	TR3
Find_Next_Instance		X	X		X		X	TR3
CM ASE								
CM_Open		X	X		X		X	
CM_Close		X	X		X		X	
CM_Unconnected_Send		X	X		X		X	
AR ASE								
UCMM_Create		X	X					
UCMM_Delete	X		X					
UCMM_Write		X	X		X		X	
UCMM_Abort	X		X		X			
TR_Write	X		X					TR0 – TR6
TR_Trigger	X		X					TR0 – TR3
TR_Verify	X		X		X			TR3, TR6
TR_Packet_Received	X				X			TR0 – TR6
TR_Ack_Received	X		X					TR2
TR_Status_Update	X		X		X			TR4, TR5

9 Communication Model Type 3 Specification

9.1 Type 3 Concepts

9.1.1 User Requirements of the Type 3 DP System

The typical automation system consists of one or several programmable controllers connected by I/O-systems to the machine/process. The I/O-system is mostly implemented as a set of single point connections with a well defined electrical interface such as 24V digital signals or 4 to 20mA analogue signals. The overall structure of the programmable controller is defined in IEC 61131-1 and widely used.

To switch from I/O-systems plugged in a programmable controller to decentralised I/O-systems a serial link is necessary not only for the exchange of process data, but also to transmit diagnosis, configuration data and parameters.

Furthermore many applications in factory automation require very short reaction times.

Therefore the user interface in the programmable controller for the exchange of process data should be designed in a way to allow direct access to these variables without specific function calls.

In addition a reliable data transfer is required for industrial applications.

The integration of simple field devices requires a simple protocol architecture to allow low cost implementations.

To support the integration of smart field devices acyclic communication services and alarms are needed.

In order to achieve an automation system consisting of devices from different manufacturers strict interoperability of all components is necessary.

In accordance with the IEC 61131-1 structure the decentralised structure should be invisible to an existing control program.

9.1.2 Features of Type 3 DP

Combining the structure defined in IEC 61131-1 with Type 3 DP 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 devices (sensor/actuator) and to/from the programmable controller over long distances via Type 3 DP with high accuracy
- Parameterisation and diagnosis can be done without additional wiring
- Commissioning of subparts without additional installations

Table 113 gives an overview of the requirements and features of a Type 3 DP system.

Table 113 – Requirements and Features of Type 3 DP

Requirement	Feature of Type 3 DP
Short reaction time	Exchange of more than 1000 Inputs and Outputs with 32 field devices in less than 10 ms
Automation system consists of one or several programmable controller	Mono-Master or Multi-Master Operation
Simple field devices	Simple protocol, low cost communication interface
Reliable data transfer	Hamming Distance 4 guaranteed; Error recovery mechanism; Use of proven transmission technology;
Excellent diagnosis	Various diagnosis in Masters as well as in Slaves
User interface with direct access to process data	Buffer interface for process data
Smart field devices	Acyclic communication which provides a flexible and enhanced addressing scheme of data within field devices. The possibility to transfer an alarm from the field device to the programmable controller with an explicit acknowledgement.
Interoperability	Precise and complete definitions including the definition of the system behaviour.

9.1.3 DP Communication Model

The DP communication model supports the communication of field devices with a single controlling device (e.g. programmable controller or distributed control system) via associations (see Figure 38). Furthermore, the communication between several controlling devices and their assigned field devices is also supported (see Figure 39).

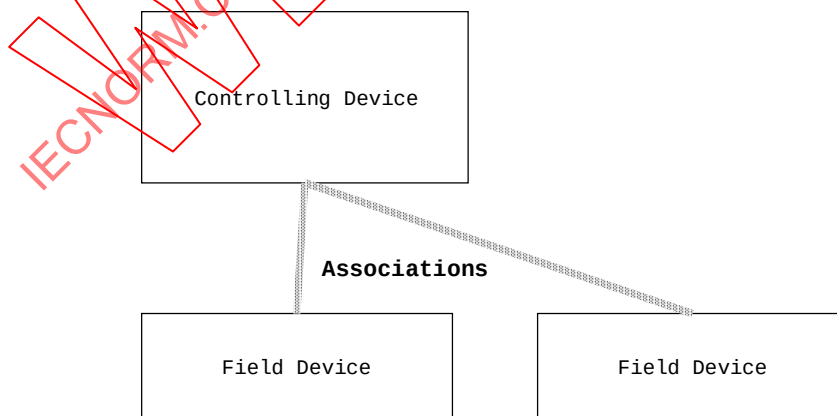


Figure 38 – Example of DP Communication with a single Controlling Device

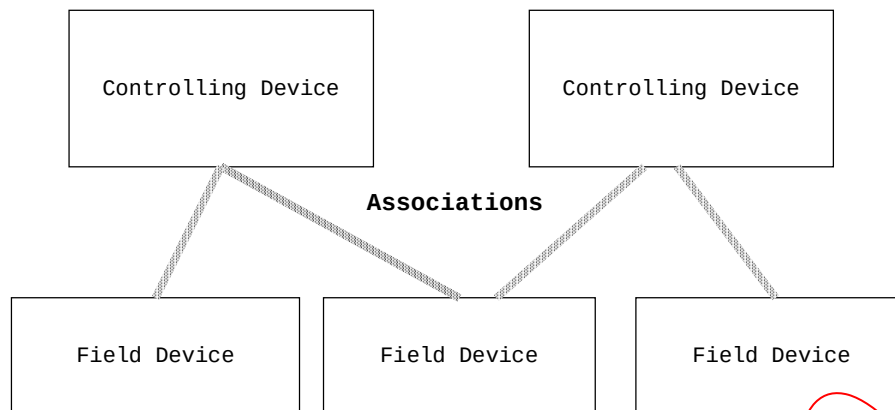


Figure 39 – Example of DP Communication with several Controlling Devices

9.1.3.1 Device Types

Type 3 DP defines three main device types.

9.1.3.1.1 DP-Master (Class 1)

The DP-Master (Class 1) is a controlling device which is associated with one or more DP-Slaves (field devices). The DP-Master (Class 1) performs one or more of the following basic functionalities:

- cyclic exchange of I/O data with related DP-Slaves
- diagnosis with the DP-Slaves
- configuration of DP-Slaves
- treatment of configuration and diagnosis requests of a DP-Master (Class 2)

The extended functionalities of a DP-Master (Class 1) are:

- acyclic access to process data of the DP-Slaves
- treatment of alarms from DP-Slaves

9.1.3.1.2 DP-Master (Class 2)

The DP-Master (Class 2) is a controlling device which manages provision of configuration data (parameter sets) and collection of Diagnosis data for/from a DP-Master (Class 1). Additionally the DP-Master (Class 2) can perform all basic and some extended communication capabilities of a DP-Master (Class 1) to a DP-Slave.

9.1.3.1.3 DP-Slave

The DP-Slave is a field device and performs the following activities depending on the functionality.

The basic functionalities are:

- cyclic exchange of I/O data with the assigned DP-Master
- diagnosis with the assigned DP-Master
- treatment of configuration requests of a DP-Master

The extended functionalities of a DP-Slave are:

- provision of acyclic access to process data for DP-Masters
- provision of alarms to the assigned DP-Master

DP-Slaves are composed of one or more modules. These modules usually reflect the hardware components of the field device.

9.1.3.1.3.1 Module

The DP Application Layer uses modules to reflect the structuring of components (e.g. hardware units, logical units) inside a DP-Slave.

Each module will be addressed by a slot number (1 to 254). Numbering is without gaps, ascending, beginning with 1. If a slot is not occupied with a module, an empty slot will be registered under the corresponding slot number in the configuration. The slot number "zero" (Slot = 0) refers to the DP-Slave itself. Modules contain data which is index addressable. The concrete use of slot number and index to address data within the device is manufacturer specific.

A module provides one or several of the following data elements:

- Configuration
- Diagnosis
- I/O Data
- Alarm
- Process Data

In modular DP-Slaves modules determine the actual configuration of the field device (see Figure 40). The number of modules is fixed according to the device specification (maximum number of modules is 244).

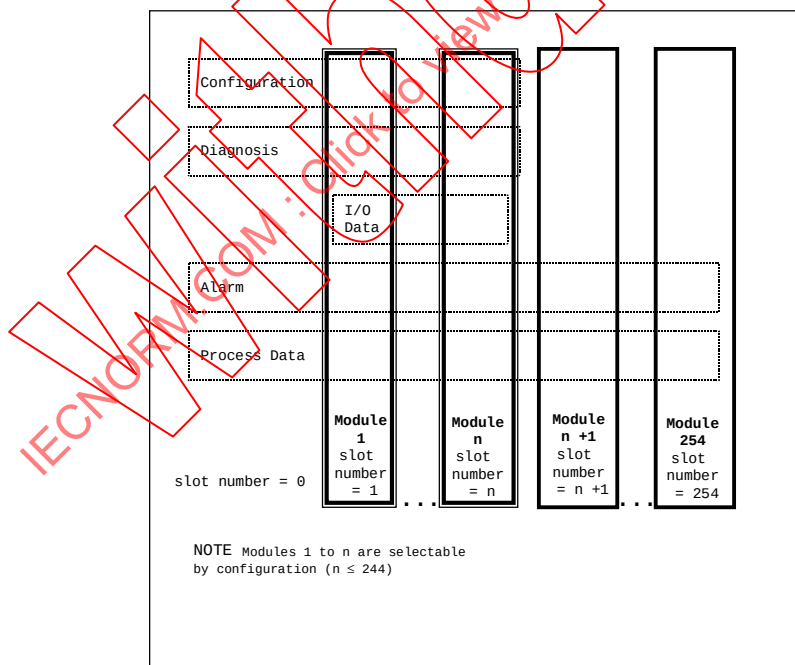


Figure 40 – DP-Slave Model (modular DP-Slave)

In compact DP-Slaves with respect to the device specification the number of modules is fixed and therefore also the configuration (for example only one module, see Figure 41).

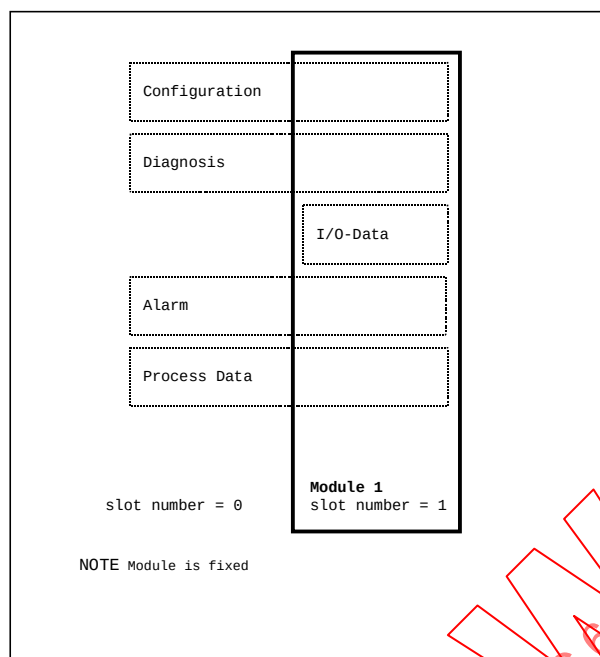


Figure 41 – DP-Slave Model (compact DP-Slave)

9.1.3.2 Device Attributes

Device attributes indicate that besides the basic functionality, extended functionality will be supported additionally.

These additional functionalities can imply certain rules for already defined data structures. A device attribute indicates that the device supports additional communication relationships or different communication behaviour.

9.1.3.2.1 Device Attribute DPV1

With the device attribute DPV1 the following conventions are implied for a Type 3 DP device:

- If a DP-Slave supports the functionality combined with the attribute DPV1 the keyword "DPV1_Slave" specified in the GSD has to be set accordingly.

The format of the Device Related Diagnosis depends on the attribute DPV1:

- DPV1 = TRUE means that this DP-Slave shall not use the Device Related Diagnosis
- DPV1 = FALSE means that this DP-Slave may use the Device Related Diagnosis

The parameterisation data of a DP-Slave contains Device and/or Module Related User Parameter. The possible length of the Device and/or Module Related User Parameter depends on the attribute DPV1:

- DPV1 = TRUE indicates that the first three octets of the Device and/or Module Related User Parameter has to be used to implement the extended parameterization functionality.
- DPV1 = FALSE indicates that Device and/or Module Related User Parameter are used to implement the basic parameterization functionality.

A device with the attribute DPV1 set to TRUE can support the following extended communication functionalities:

- acyclic access to process data
- transfer of alarms

These extensions are optional, see Table 114. It is guaranteed that devices with basic communication functionalities as well as devices with these extended functionalities can operate on the same network and are interoperable.

Therefore:

- a DP-Slave with extended functionalities can be operated also on a DP-Master only with the basic functions
- a DP-Slave only with the basic functions can operate without restrictions on a DP-Master with the extended functionalities.

Table 114 – Overview of the different functionality of DP-Devices

Device types	basic functionality	extended functionality acyclic services (DPV1 = TRUE)	alarms (DPV1 = TRUE)	further functions
DP-Master (Class 1)	mandatory	optional	optional	
DP-Master (Class 2)	mandatory	optional	not supported	
DP-Slave	mandatory	optional	optional	

Table 115 makes obvious that a device which supports alarms has also to support the acyclic services.

Table 115 – Valid combinations of supported extended functionalities

extended functionality	attribute DPV1 = FALSE	attribute DPV1 = TRUE					
Device Related Diagnosis	a.	n. a.	n. a.	n. a.	n. a.	n. a.	n. a.
extended parameterization	n. a.	s.	s.	s.	s.	s.	s.
alarm handling	n. a.	n. s.	n. s.	n. s.	n. s.	s.	s.
acyclic services (DP-Master (Class 1))	n. a.	n. s.	n. s.	s.	s.	s.	s.
acyclic services (DP-Master (Class 2))	n. a.	n. s.	s.	s.	n. s.	n. s.	s.
where n. a.: not allowed a.: allowed s.: supported n. s.: not supported							

9.1.3.3 Application Process

In the fieldbus 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)(see Figure 42). A device may have several APs. In this case each individual AP is uniquely identified by an AP Identifier (API).

Every DP-Slave shall have a Default AP with API = 0. The Default AP may be related to I/O data, to diagnosis and alarms, to configuration and to process data. All other DP-Slave APs shall be related to process data only.

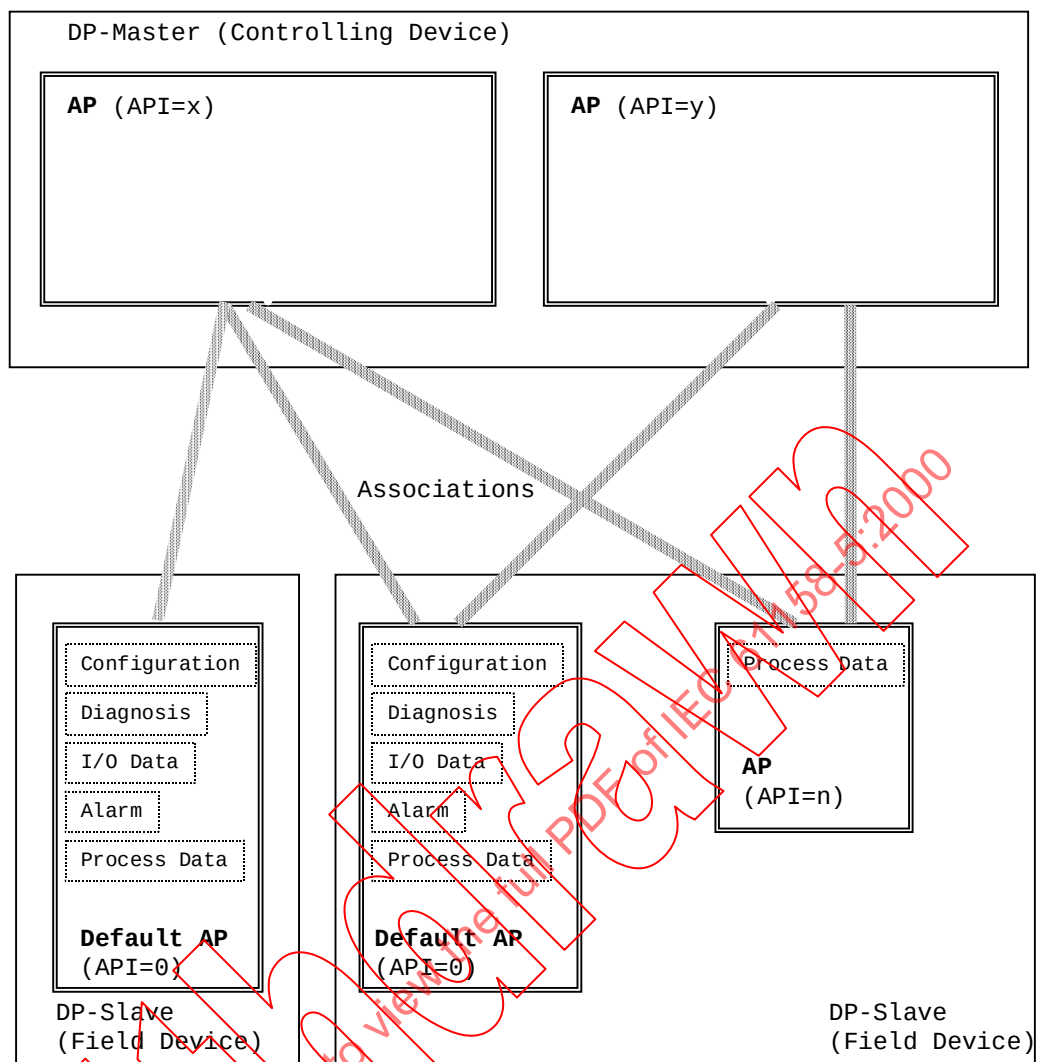


Figure 42 – Overview of Application Processes

An AP may be distributed to several modules. Each module within an application process is identified uniquely by a slot number (Slot = 1 to 254). The slot number "zero" (Slot = 0) refers to the AP itself within the field device.

The Configuration data shall be related to the Default AP itself and to all its modules containing I/O Data.

The Diagnosis information may only be related to the Default AP itself and to all its modules containing I/O Data.

The I/O Data shall be related to modules of the Default AP and not to the AP itself. The number of modules with I/O Data is limited and depends on the configuration and I/O Data structure of the modules (see clause 0).

The Alarm data may be related to the Default AP itself and to all its modules.

The Process Data may be related to all APs and to all their modules.

Figure 43 shows the relationship between the APs, the data elements and the modules of a DP-Slave.

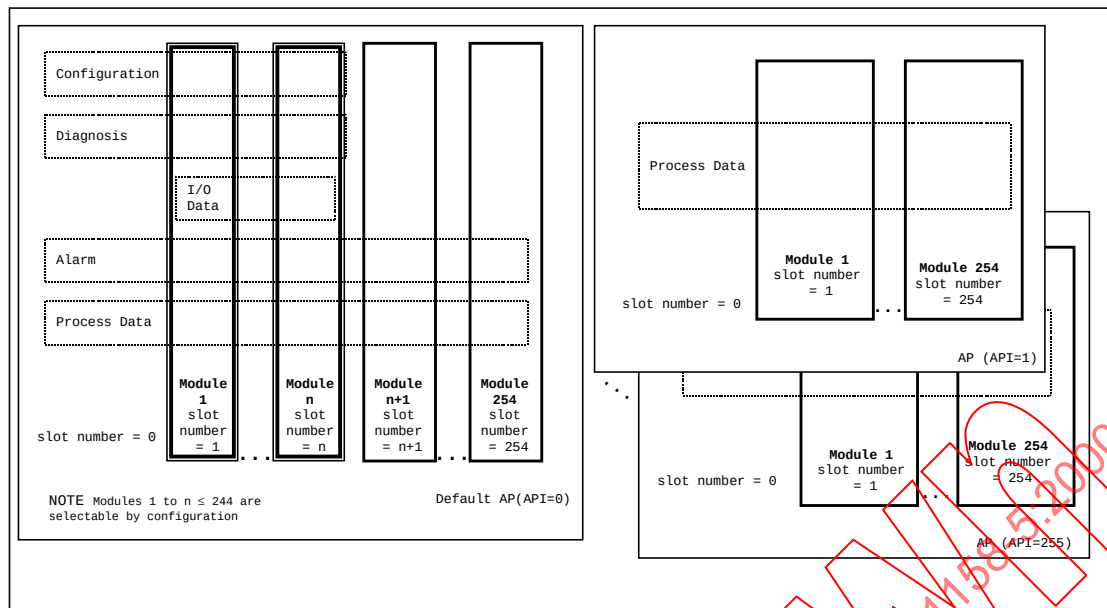


Figure 43 - DP-Slave Model (modular DP-Slave)

The DP-Master (Class 1) shall have only its Default AP. The DP-Master (Class 1) shall communicate with the Default AP of a DP-Slave only.

The DP-Master (Class 2) may have several APs. Communicating with a DP-Slave the DP-Master (Class 2) may be able to address all APs of a DP-Slave.

9.1.3.3.1 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 DP Application Layer offers the following ASEs (see Figure 44):

I/O Data ASE

The I/O Data ASE provides a set of services to convey I/O data cyclically. These data always belong to those modules that have been configured in terms of the Configuration ASE. A multicast service provides the synchronisation of the I/O data and reports the operational state of the assigned DP-Master.

Process Data ASE

The Process Data ASE provides a set of services to convey acyclically data. The application of the DP-Master requests each transmission individually. The Process Data ASE can be related to all APs of a DP-Slave.

Diagnosis ASE

The Diagnosis ASE provides services for the DP-Master to fetch Diagnosis information from a DP-Slave which is related to the Default AP or to its modules that are part of the configuration.

Alarm ASE

The Alarm ASE provides a set of services to convey alarms or status messages issued by the DP-Slave. The assigned DP-Master acknowledges the Alarms. Status messages will not have to be acknowledged by the DP-Master.

Management ASE

The Management ASE provides a set of services to manage a DP-Master (Class 1) by a DP-Master (Class 2). The DP-Master (Class 1) receives the bus parameter set or the slave parameter set. Afterwards the received sets will be activated by the DP-Master (Class 2). The DP-Master (Class 1) will provide the system diagnosis information if requested.

Context ASE

The Context ASE provides a set of services to

- parameterise and configure the Default AP itself and its modules containing I/O data
- set the device address of a DP-Slave
- to provide the real configuration of the DP-Slave to be read by a DP-Master (Class 2)
- and to establish or release an association between an individual AP of a DP-Master (Class 2) and an individual AP of the DP-Slave for acyclic communication. This association exclusively belongs to the Process Data ASE.

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.

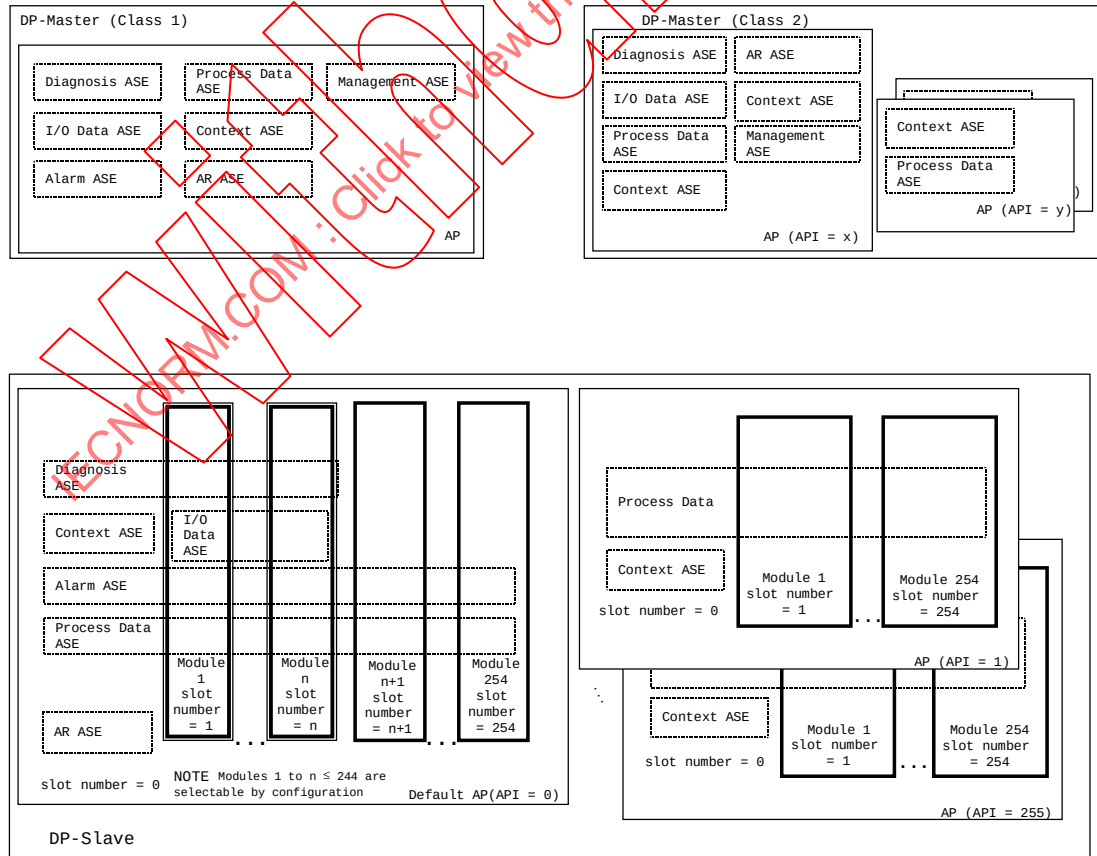


Figure 44 – Application Service Elements (ASEs)

9.1.3.3.2 Application Objects

An application object (APO) is a network representation of a specific aspect of an AP. Each APO represents a set of information and processing capabilities of an AP that are accessible through services of the DP Application Layer. APOs are used to represent these capabilities to other APs in a Fieldbus system.

In order to permit an AP to communicate with an AP of an another device, APOs have to be available. Application objects (virtual objects) represent existing process objects (real objects), that an application process has made visible and accessible to the communication (see Figure 45).

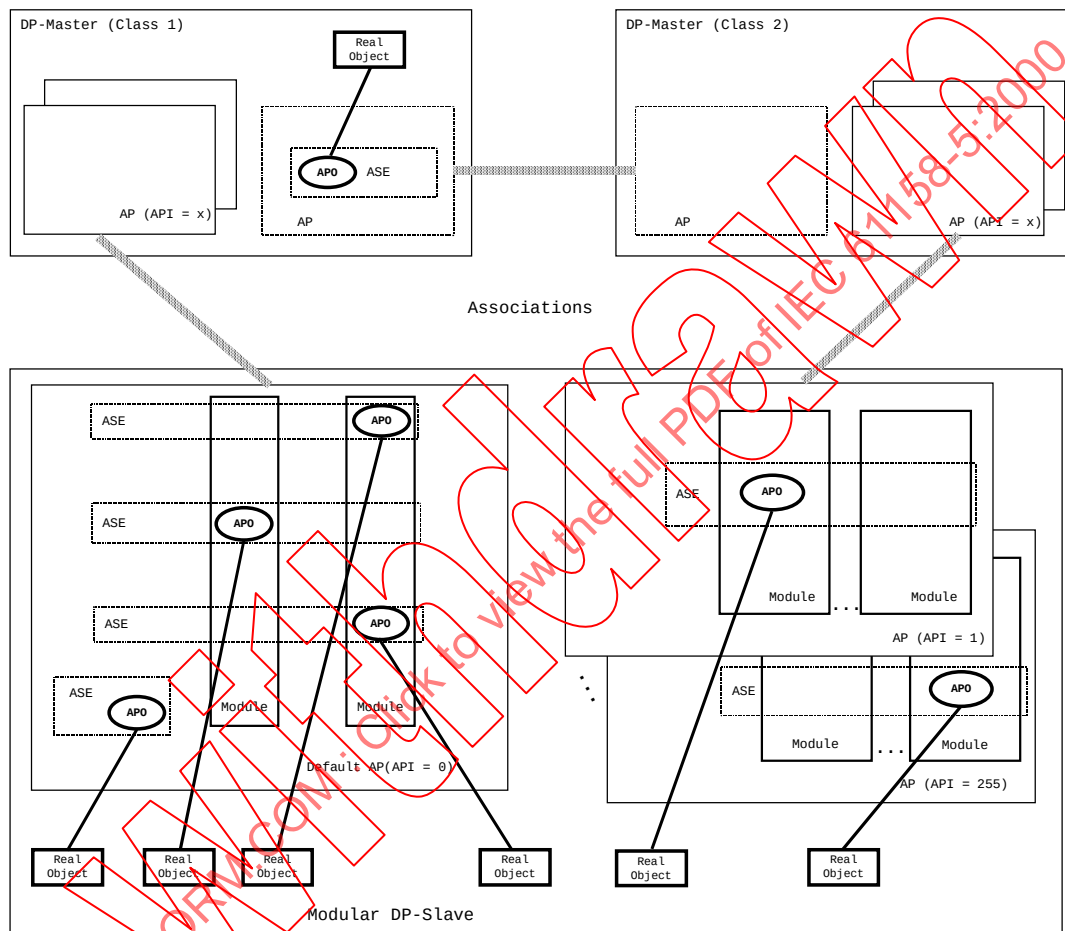


Figure 45 – Application Process with Application Objects (APOs)

In Figure 46, 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 and index. The address space which is defined by slot and index can be used by several APs.

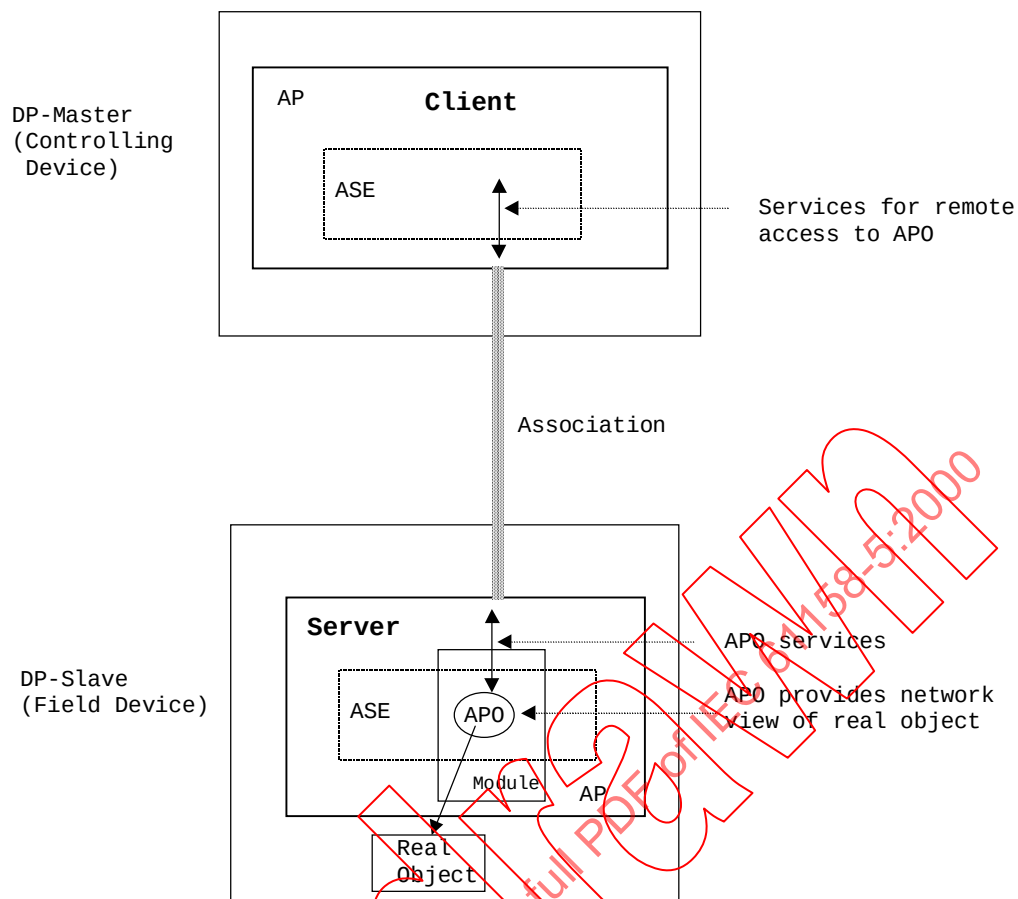


Figure 46 – Access to a remote APO

9.1.3.4 Application Relationship

9.1.3.4.1 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 47). This relationship is activated by the exchange of Application Protocol Data Units (APDU). Type 3 DP uses different types of ARs which distinguish in their conveyance characteristic.

9.1.3.4.2 Application Relationship Endpoint

An AP has access to the communication using application relationship endpoints (AREP, see ISO 7498). 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 47 shows an example of one AR with two AREPs.

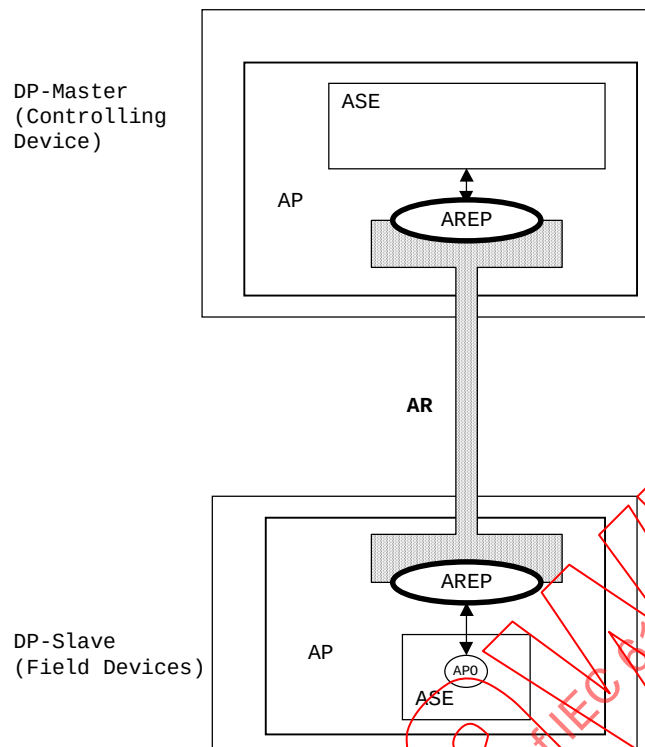


Figure 47 – Example of one AR with two AREPs

9.1.3.4.3 Overview of Application Relationships

The DP Application Layer offers the following AR types:

MS0: application relationship between the application processes of one DP-Master (Class 1) and optional the AP of one or several DP-Master (Class 2) with all related DP-Slaves for the following purposes:

- cyclic exchange of the I/O data (DP-Master (Class 1))
- acyclic data transfer for parameterisation, configuration and diagnosis (DP-Master (Class 1))
- acyclic transfer of commands to a set of field devices (DP-Master (Class 1))
- acyclic read of the I/O data (DP-Master (Class 2))
- acyclic read of configuration information (DP-Master (Class 2))
- acyclic read of diagnosis information (DP-Master (Class 2))
- acyclic write of remanent parameter (DP-Master (Class 2))

MS1: connection-oriented relationship between the application processes of one DP-Master (Class 1) and of one related DP-Slave for the following purposes:

- acyclic read and write of variables
- acyclic transfer of alarms

MS2: connection-oriented relationship between the application processes of one DP-Master (Class 2) and of one related DP-Slave for the purpose:

- acyclic read and write of variables

MM1: connectionless relationship between the application processes of one configuration device (Master Class 2) and of one related controlling device (Master Class 1) for the following purposes:

- upload and download for configuration information
- upload of diagnosis information
- activation of the previously transferred configuration

MM2: connectionless relationship between the application processes of one configuration device (Master Class 2) and of a set of related controlling devices (Master Class 1) for the purpose:

- activation of the previously transferred configurations

9.1.4 Dynamical Behaviour of Type 3 DP

Before commissioning a DP system, all stations shall have been assigned an unique address. In the case of DP-Slaves, this address may be set up using the MS0 AR (see 9.2.5.3.7).

All DP-Slaves that have not yet been assigned an individual address, shall start with the default address 126. Only one device with this address shall be allowed on the network. A DP-Master address shall not be set to the default address. A DP-Master (Class 2) may access a DP-Slave via the default address and may assign a specific address to that device.

For security reasons, the DP-Master (Class 1) shall not exchange I/O data with a DP-Slave whose address is 126.

To exchange I/O data a DP-Master (Class 1) shall have a valid master parameter set which may be loaded and activated by the DP-Master (Class 2) via the MM1 AR and MM2 AR. The master parameter set consists of several sets of configuration data and parameters, each dedicated to a DP-Slave. This set of data shall contain all necessary information (for the DP system) to describe a DP-Slave. Additionally, the master parameter set shall include the bus parameters as well as the address assignment table which assigns each individual remote I/O data a unique address in the I/O space of the DP-Master's User.

If there exists a valid master parameter set in a DP-Master (Class 1), this device starts to check whether the DP-Slaves dedicated to that DP-Master are present or not. After an appropriate answer the DP-Master (Class 1) will set the parameters and check the configuration of that DP-Slave. A DP-Slave will accept a check configuration request only from the Master who sets the parameters.

After submitting these two functions correctly, a DP-Master (Class 1) checks the status of the DP-Slave by reading the Diagnosis information. If the configuration check was successful and parameterization was correct, the DP-Master (Class 1) will enter the user data exchange mode. A DP-Slave accepts data exchange requests only from a DP-Master (Class 1) which has previously submitted parameters and configuration. If a diagnosis event occurred in the application process of a DP-Slave, this event is signalled by sending a response with high priority. The DP-Master (Class 1) will read the diagnosis and inform its local User.

Only in the data exchange mode the extended functionalities like the acyclic read and write of variables and the acyclic transfer of alarms are possible if supported by both the DP-Master (Class 1) and the DP-Slave. Errors on the acyclic communication (MS1 AR) also influence the cyclic data transfer (MS0 AR) and vice versa. This means that both communication relationships will be aborted. Each alarm has to be acknowledged by the assigned DP-Master (Class 1).

Every DP-Master (Class 2) can read the diagnosis information, the configuration, the input and output data (via the MS0 AR) of every DP-Slave although the DP-Slave is under control of a DP-Master (Class 1). A DP-Master (Class 2) can take a DP-Slave under its control. In this case the DP-Slave will stop the data exchange mode with the DP-Master (Class 1). In this case the DP-Master (Class 1) starts to read the diagnosis cyclically. If the DP-Master (Class 2) has finished its

communication with this DP-Slave, the DP-Slave will set the field "Master Add(ress)" to invalid. This event triggers the DP-Master (Class 1) to gain control over this DP-Slave again. As described above it will send parameters and configuration first.

If supported by both the DP-Master (Class 2) and the DP-Slave extended functionality like the acyclic read and write of variables is possible (via the MS2 AR). The MS2 AR is independent of the MS0 AR and MS1 AR.

9.1.5 Type 3 Status Values

Table 116 – Status Values of the service primitives

Short Name	Meaning
AD	Access denied
DI	Data Incomplete
DS	Local-DL/PHY Entity is not in logical token-ring or disconnected from line
EA	Area too large (Up-/Download)
FE	Format-Error in a Request-frame
IP	Invalid Parameter
IV	Invalid parameters in request
LE	Data-block-length too large (Up-/Download)
NA	Negative ack, no reaction from remote station
NC	Master parameter set not compatible
NE	Area non-existent
NI	Service not implemented
NO	Service in this state not possible
NR	No Response data
RE	Format-Error in a Response-frame
RR	Resources of the remote-DL Entity not sufficient or not available
RS	Service or remote-address at remote-LSAP or remote-LSAP not activated - remote-station is no DP-Station - remote-station is not yet ready for these Service - remote-station is associated with an other Requester - optional service not available
SC	Sequence Conflict
SE	Sequence Error
TO	Function-Timeout expired
UE	Remote-DMPM / DL interface error
DH	Positive acknowledgement for sent data, reply data with high priority available
LR	DL/DLM_status: Local resource not available or not sufficient
RDL	DL/DLM_status: Response DL/DLM-data low and no resource for send data

(continued on next page)

RDH	DL/DLM_status: Response DL/DLM-data high and no resource for send data
ABT_SE	sequence error; service not allowed in this state
ABT_FE	invalid request PDU received
ABT_TO	timeout of the connection
ABT_RE	invalid response PDU received
ABT_IV	invalid service from the User
ABT_STO	Send Timeout requested was too small
ABT_IA	invalid additional address information
ABT_OC	waiting for DL-DATA-REPLY.con

9.2 Type 3 ASEs

9.2.1 Process Data ASE

9.2.1.1 Overview

In the DP Application Layer environment, application processes contain data that remote applications are able to read and write. The Process Data ASE defines attributes of general purpose Process Data objects and provides a set of services used to read, and write their values. A Process Data ASE is specific for an AP and every AP may contain one Process Data ASE. Process Data objects within an AP are addressed by Slot Numbers and Indices. The Process Data object can be read and written partially or entirely.

The access to Process Data objects is performed according to the client/server access model. The client/server model is characterised by a client application sending a read or write request to a server application that responds accordingly. The server's activity is stimulated by clients on the network.

The formal model of the Process Data ASE is presented by the Process Data class specification, containing a description of its attributes, services as well as invocations, followed by a detailed service specification.

9.2.1.2 Process Data Class Specification

The Process Data ASE defines two Process Data object types:

- Simple objects
- Combined objects

Simple objects may have a one to one relation to real objects. Combined objects may have a one to two relation to real objects. Combined objects may be read and written with one service interaction. This type of objects may be used for communication which requires consistency between read and write or high protocol efficiency.

9.2.1.2.1 Simple Process Data Class Specification

A Simple Process Data object is always related to one real object as shown in Figure 48.

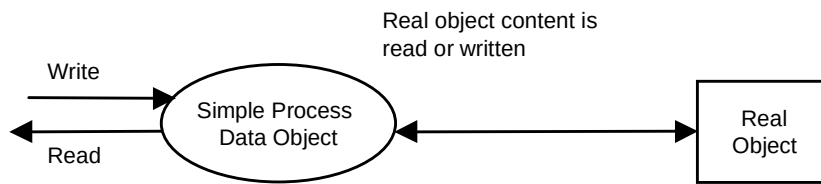


Figure 48 – Relation of a Simple Process Data object to the real object

A Process Data object is described by the following template:

DP ASE:	Process Data ASE
CLASS:	Simple Process Data
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1.	(m) Key Attributes: Identifier
2.	(m) Attribute: Process Data Description
3.	(m) Attribute: Partial Access
4.	(m) Attribute: Access Rights MS1
5.	(m) Attribute: Access Rights MS2
6.	(c) Constraint: Access Rights = Right to Read
6.1.	(m) Attribute: List of SCL Read
6.1.1.	(m) Attribute: SCL
7.	(c) Constraint: Access Rights = Right to Write
7.1.	(m) Attribute: List of SCL Write
7.1.1.	(m) Attribute: SCL
8.	(o) Attribute: Process Data Name
9.	(o) Attribute: Local Detail
SERVICES:	
1.	(o) OpService: Read
2.	(o) OpService: Write

9.2.1.2.1.1 Attributes

Identifier

This key attribute is composed of API, Slot Number and Index to define to which Application Process Identifier, module and to which index within the specified module the Simple Process Data object belongs.

Attribute Type: Unsigned8, Unsigned8, Unsigned8

Allowed values: 0 to 254, 0 to 254, 0 to 254

Process Data Description

This attribute refers to the data description of the Process Data ASE. For data description the template given in 5.2.1 is used.

Partial Access

This attribute defines whether a partial access to the Simple Process Data object is supported or not. Partial access equal TRUE means that the content of the Simple Process Data object is read or written from the first octet up to the octet defined by the length given in the service. The length given in the service shall be less or equal the value of the attribute length in case of a write access. For a read access the entire object will be read if the length given in the service is greater than the length of the object. Partial access equal FALSE means that the content of the Simple Process Data object can only be read or written entirely. That implies that the length given in the service shall be equal the length of the object.

Attribute Type: Boolean

Access Rights MS1

This attribute defines the type of access rights defined for a MS1 AR for the Simple Process Data object.

Allowed values:

Table 117 – Access Rights MS1

Value	Meaning
0	Right to Write
1	Right to Read
2	Right to Read/Write
3	No Access

Access Rights MS2

This attribute defines the type of access rights defined for a MS2 AR for the Simple Process Data object.

Allowed values:

Table 118 – Access Rights MS2

Value	Meaning
0	Right to Write
1	Right to Read
2	Right to Read/Write
3	No Access

List of SCL Read

This attribute is composed of the following list elements:

SCL

The attribute SCL contains one value for read access protection to this Simple Process Data object. The rules for accepting or rejecting a read access are described in subclause 9.2.1.3.

Attribute Type: Unsigned8

List of SCL Write

This attribute is composed of the following list elements:

SCL

The attribute SCL contains one value for write access protection to this Simple Process Data object. The rules for accepting or rejecting a write access are described in subclause 9.2.1.3.

Attribute Type: Unsigned8

Process Data Name

This optional attribute defines the name of the Process Data object.

Attribute Type: Visible String(32)

Local Detail

This optional attribute contains additional local information about the Simple Process Data object.

Attribute Type: Octet String

9.2.1.2.1.2 Invocation of the Simple Process Data object

For the invocation of the Simple Process Data object the following rules have to be considered:

- Simple Process Data objects shall not exceed the total length of 240 octets. The attributes Data Type, Number of Elements and Length have 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).

9.2.1.2.2 Combined Process Data Class Specification

A Combined Process Data object is always related to two real objects as shown in Figure 49.

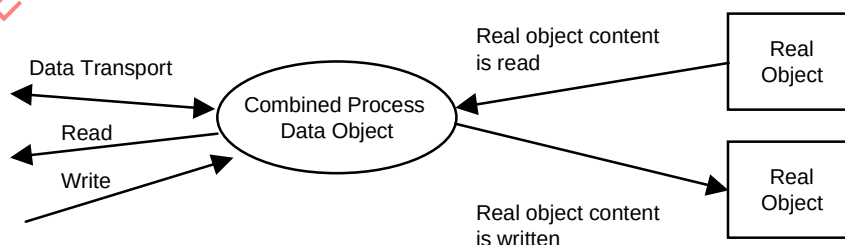


Figure 49 – Relation of a Combined Process Data object to the real objects

A Combined Process Data object is described by the following template:

DP ASE:	Process Data ASE
CLASS:	Combined Process Data
CLASS ID:	not used
PARENT CLASS:	TOP

ATTRIBUTES:

1. (m) Key Attributes: Identifier
2. (m) Attribute: Read Process Data Description
3. (m) Attribute: Read Partial Access
4. (c) Constraint: Access Rights = Right to Read
 - 4.1. (m) Attribute: List of SCL Read
 - 4.1.1. (m) Attribute: SCL
5. (m) Attribute: Write Process Data Description
6. (m) Attribute: Write Partial Access
7. (c) Constraint: Access Rights = Right to Write
 - 7.1. (m) Attribute: List of SCL Write
 - 7.1.1. (m) Attribute: SCL
8. (m) Attribute: Access Rights MS1
9. (m) Attribute: Access Rights MS2
10. (o) Attribute: Process Data Name
11. (o) Attribute: Local Detail

SERVICES:

1. (o) OpService: Data Transport
2. (o) OpService: Read
3. (o) OpService: Write

9.2.1.2.2.1 Attributes**Identifier**

This key attribute is composed of API, Slot Number and Index to define to which Application Process Identifier, module and index within the specified module the Combined Process Data object belongs.

Attribute Type: Unsigned8, Unsigned8, Unsigned8

Allowed values: 0 to 254, 0 to 254, 0 to 254

Read Process Data Description

This attribute refers to the data description of the Process Data ASE. For data description the template given in 5.2.1 is used.

Read Partial Access

This attribute defines whether a partial read access to the Combined Process Data object is supported or not. Partial access equal TRUE means that the content of the Combined Process Data object is read from the first octet up to the octet defined by the length given in the service. The entire object will be read if the length given in the service is greater than the length of the object. Partial access equal FALSE means that the content of the Combined Process Data object can only be read entirely. That implies that the length given in the service shall be equal the length of the object.

Attribute Type: Boolean

List of SCL Read

This attribute is composed of the following list elements:

SCL

The attribute SCL contains one value for read access protection to this Combined Process Data object. The rules for accepting or rejecting a read access are described in subclause 9.2.1.3.

Attribute Type: Unsigned8

Write Process Data Description

This attribute may contain either a Simple Data Description or an Array Data Description or a Record Data Description.

Write Partial Access

This attribute defines whether a partial write access to the Combined Process Data object is supported or not. Partial access equal TRUE means that the content of the Combined Process Data object is written from the first octet up to the octet defined by the length given in the service. The length given in the service shall be less or equal the value of the attribute length. Partial access equal FALSE means that the content of the Combined Process Data object can only be written entirely. That implies that the length given in the service shall be equal the length of the object.

Attribute Type: Boolean

List of SCL Write

This attribute is composed of the following list elements:

SCL

The attribute SCL contains one value for write access protection to this Combined Process Data object. The rules for accepting or rejecting a write access are described in subclause 9.2.1.3.

Attribute Type: Unsigned8

Access Rights MS1

This attribute defines the type of access rights defined for a MS1 AR for the Combined Process Data object.

Allowed values:

Table 119 – Access Rights MS1

Value	Meaning
0	Right to Write
1	Right to Read
2	Right to Read/Write
3	No Access

Access Rights MS2

This attribute defines the type of access rights defined for a MS2 AR for the Combined Process Data object.

Allowed values:

Table 120 – Access Rights MS2

Value	Meaning
0	Right to Write
1	Right to Read
2	Right to Read/Write
3	No Access

Process Data Name

This optional attribute defines the name of the Process Data object.

Attribute Type: Visible String(32)

Local Detail

The attribute local detail contains additional local information about the Simple Process Data object.

Attribute Type: Octet String

9.2.1.2.2 Invocation of the Combined Process Data object

For the invocation of the Combined Process Data object the following rules have to be considered:

- Combined Process Data objects shall not exceed the total length of 240 octets for both Read Process Data and Write Process Data . The attributes Data Type, Number of Elements and Length have 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 optional service(s).

9.2.1.3 Access Protection on Process Data objects

The Process Data ASE in combination with the Context ASE provides also access protection. This is not a protection against intentional misuse of the communication facilities of a field device but helps to protect a system for accidental erroneous use of Process Data.

The protection is accomplished by two mechanisms:

- the Initiate service will set up a security level for this CR.
- the Process Data object will have a list of SCL values for the read and write service. An access will be accepted only if the SCL of the initiate service matches (see Table 121) with an element of the SCL list for this Process Data object.

This allows the User to select a level for read and write objects. Thus a service device can use a SCL which allows access to almost all objects. A text display may have only read access to specific objects.

Table 121 – SCL matching rules

SCL in Initiate.ind	SCL in Process Data object	Condition	Result
= 0	= 0	-	access allowed
≠ 0	= 0	-	access allowed
= 0	≠ 0	-	access denied
≠ 0	≠ 0	SCL in Initiate.ind <u>not contained in</u> List of SCL at Process Data object	access denied
≠ 0	≠ 0	SCL in Initiate.ind <u>contained in</u> List of SCL at Process Data object	access allowed

9.2.1.4 Process Data Service Specification

9.2.1.4.1 Read

This confirmed service may be used to read the value of a Process Data object. It may be used only via the MS1 or MS2 AR.

Table 122 – Read

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Slot Number	M	M(=)		
Index	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Length			M	M(=)
Data			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.

Slot Number

The parameter Slot Number is used in the destination device for addressing the desired Process Data object in a specific slot (typically a module).

Index

The parameter Index is used in the destination device for addressing the desired Process Data object.

Length

The parameter Length indicates the number of octets of a Process Data object that has to be read.

Result (+)

This selection type parameter indicates that the service request succeeded.

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 5.2.1.

Result (-):

This selection type parameter indicates that the service request failed.

Error Decode:

This parameter selects one of the following Error schemes: DPV1, FMS, Profile_Specific.

Error Code 1:

In case of Error Decode equal DPV1 the Error Code 1 assumes one of the following values: read error, write error, module failure, version conflict, feature not supported, user specific, invalid index, write length error, invalid slot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, write constrain conflict, resource busy, resource unavailable.

Error Code 2:

The parameter Error Code 2 is user specific.

9.2.1.4.2 Write

This confirmed service may be used to write the value of a Process Data object. It may be used only via the MS1 or MS2 AR.

Table 123 – Write

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Slot Number	M	M(=)		
Index	M	M(=)		
Length	M	M(=)		
Data	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Length			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.

Slot Number

The parameter Slot Number is used in the destination device for addressing the desired Process Data object in a specific slot (typically a module).

Index

The parameter Index is used in the destination device for addressing the desired Process Data object.

Length

The parameter Length indicates the number of octets of a Process Data object which has to be written.

Data

The parameter Data contains the value of the Process 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.

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

Error Decode:

This parameter selects one of the following Error schemes: DPV1, FMS, Profile_Specific.

Error Code 1:

In case of Error Decode equal DPV1 the Error Code 1 assumes one of the following values: read error, write error, module failure, version conflict, feature not supported, user specific, invalid index, write length error, invalid slot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, write constrain conflict, resource busy, resource unavailable.

Error Code 2:

The parameter Error Code 2 is user specific.

9.2.1.4.3 Data Transport

This service is used to write and read the values of a single Combined Process Data object. This service may be used only via MS2 AR.

With this service the data of a Process Data object are transferred

- from the DP-Master to the DP-Slave,
- from the DP-Slave to the DP-Master (Class 2) or
- from the DP-Master to the DP-Slave and back

Table 124 – Data Transport

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Slot Number	M	M(=)		
Index	M	M(=)		
Length	M	M(=)		
Data	C	C(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Length			M	M(=)
Data			C	C(=)
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.

Slot Number

The parameter Slot Number is used in the destination device for addressing the desired slot (typically a module).

Index

The parameter Index is used in the destination device to address the desired data.

Length

In the Request the parameter Length indicates the number of octets of the parameter "Data" that are transferred from the client to the server. In the Response the parameter Length indicates the number of bytes of the parameter "Data" that are transferred from the server to the client.

Data

The parameter Data contains the values transferred from the client to the server and vice versa and consists of the number of octets indicated in the parameter Length. This parameter shall be composed of the data types defined in 5.2.1.

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

Error Decode:

This parameter selects one of the following Error schemes: DPV1, FMS, Profile_Specific.

Error Code 1:

In case of Error Decode equal DPV1 the Error Code 1 assumes one of the following values: read error, write error, module failure, version conflict, feature not supported, user specific, invalid index, write length error, invalid slot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, write constrain conflict, resource busy, resource unavailable.

Error Code 2:

The parameter Error Code 2 is user specific.

9.2.2 I/O Data ASE

9.2.2.1 Overview

In the DP Application Layer environment, the default application process of a DP-Slave may contain I/O Data objects that shall be cyclically conveyed over the network. Therefore the I/O Data ASE defines attributes of the I/O Data objects and provides a set of services used to get and set the buffer to transport their values. The I/O Data objects are separated in 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 I/O Data of the modules.

Additional services are provided to read acyclically the values of the I/O Data objects and to indicate new values for the Input Data object and the Output Data object.

The I/O Data objects are implicitly addressed through the related services. The granularity of input or output data is according to the correspondent configuration information.

The I/O Data ASE uses the client/server access model. The client/server model is characterised 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 its 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.

It is possible to update only parts of the values of the I/O Data objects according to the given granularity regarding the structure of the objects. Furthermore, it is also possible that a DP-Slave has only either an Input Data object or an Output Data object.

Most DP-Slaves support only Byte or Word structures for the Input Data object and Output Data object. This type of devices use the Simple Input/Output Data object. However, complex devices use Floating Point values or Doubleword structures or even combinations of several data types. The extended versions of the Input/Output Data object reflect the requirements of these enhanced devices.

The values of the I/O Data objects of all or a group of DP-Slaves within a given network may be synchronised by its assigned DP-Master with help of the Global Control service.

The synchronisation of the values of the Input Data objects is achieved by the Freeze Command and the synchronisation of the values of the Output Data objects is achieved by the Sync Command.

With each Freeze Command the current value of the Input Data object in the MS0 buffer is frozen in the addressed DP-Slaves. If the server application now updates the value of the Input Data object by the Set Input service then this value will not be issued to the buffer until the next Freeze Command occurs. That means, that the addressed DP-Slaves will convey the "frozen" values of the Input Data objects. The synchronisation of the Input Data objects will be switched off by a Freeze Command with the value Unfreeze.

With the first Sync Command the current value of the Output Data object in the MS0 buffer is frozen in the addressed DP-Slaves. With each further Sync Command the previously transmitted values of the Output Data objects will be updated to the MS0 AR buffer for access with the Get Output service. The synchronisation of the Output Data objects will be switched off by a Sync Command with the value Unsync. A detailed description is given in clause 9.2.7.

The DP-Master (Class 1) sends information about its own operation mode to its assigned DP-Slaves for synchronisation. This will be done at every change of the operation mode of the DP-Master (Class 1) or occurs in a time interval which can be parameterised. If the DP-Master's state is CLEAR then the values of the Output Data object shall be set to a value corresponding to a safe state.

The DP-Master (Class 1) communicates via the MS0 AR with all Input Data and Output Data objects of the assigned DP-Slaves simultaneously. Additionally the client application may communicate with an Input Data object, an Output Data object, an Input Data Element or an Output Data Element of a single DP-Slave.

The DP-Master (Class 2) communicates via the MS0 AR with the Input Data and Output Data object of a single DP-Slave.

The formal model of the I/O Data ASE is presented next, followed by a description of its services.

9.2.2.2 I/O Data Class Specification

9.2.2.2.1 Simple Input Data Class Specification

The Simple Input Data object is described by the following template:

DP ASE:	I/O Data ASE
CLASS:	Simple Input Data
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m) Key Attribute:	Implicit
2. (m) Attribute:	List of Input Data Elements
2.1. (m) Attribute:	Slot Number
2.2. (m) Attribute:	Simple Input Data Description
2.2.1. (m) Attribute:	Length
2.2.2. (m) Attribute:	Format
2.2.3. (m) Attribute:	Consistency
2.2.4. (o) Attribute:	Name
SERVICES:	
1. (m) OpService:	Set Input
2. (o) OpService:	Read Input
3. (m) OpService:	Get Input
4. (o) OpService:	New Input
5. (m) OpService:	Global Control

9.2.2.2.1.1 Attributes

Implicit

The attribute Implicit indicates that the Simple Input Data object is implicitly addressed by the services.

List of Input Data Elements

This attribute is composed of the following list elements:

Slot Number

This attribute defines to which module the Input Data Element belongs.

Attribute Type: Unsigned8

Allowed values: 1 to 244

Simple Input Data Description

This attribute consists of the following attributes.

Length

The attribute Length defines the number of octets of the Input Data Element. The value of this attribute is determined by the corresponding configuration data.

Attribute Type: Unsigned8

Allowed values: 1 to 128

Format

This attribute defines the format of the Input Data Element. The value of this attribute is determined by the corresponding configuration data.

Allowed values:

Table 125 – Format

Value	Meaning
0	Byte
1	Word

Consistency

This attribute defines the Consistency of the Input Data Element. The value of this attribute is determined by the corresponding configuration data.

Allowed values:

Table 126 – Consistency

Value	Meaning
0	Consistency over the defined Format of the Input Data Element
1	Consistency over the whole Input Data Element

Name

This attribute defines the name of the Input Data Element. The value of this attribute is determined by the corresponding configuration data.

Attribute Type: Visible String(32)

9.2.2.2.1.2 Invocation of the Simple Input Data object

For the invocation of the Simple Input Data object the following rules have to be considered:

- Simple Input Data objects shall not exceed the total length of 244 octets. The attributes List of Input Data Elements and Length have to be set accordingly.
- Only one Simple Input Data object shall be invoked in a DP-Slave.
- A Simple Input Data object shall only be invoked if no Extended Input Data object (see 9.2.2.2.3) has been invoked.

9.2.2.2.2 Simple Output Data Class Specification

The Simple Output Data object is described by the following template:

DP ASE: I/O Data ASE
CLASS: Simple Output Data
CLASS ID: not used
PARENT CLASS: TOP

ATTRIBUTES:

1. (m) Key Attribute: Implicit
2. (m) Attribute: List of Output Data Elements
- 2.1. (m) Attribute: Slot Number
- 2.2. (m) Attribute: Simple Output Data Description
- 2.2.1. (m) Attribute: Length
- 2.2.2. (m) Attribute: Format
- 2.2.3. (m) Attribute: Consistency
- 2.2.4. (o) Attribute: Name

SERVICES:

1. (m) OpService: Set Output
2. (o) OpService: Read Output
3. (m) OpService: Get Output
4. (o) OpService: New Output
5. (m) OpService: Global Control

9.2.2.2.1 Attributes

Implicit

The attribute Implicit indicates that the Simple Output Data object is implicitly addressed by the services.

List of Output Data Elements

This attribute is composed of the following list elements:

Slot Number

This attribute defines to which module the Output Data Element belongs.

Attribute Type: Unsigned8

Allowed values: 1 to 244

Simple Output Data Description

This attribute consists of the following attributes.

Length

The attribute Length defines the number of octets of the Output Data Element. The value of this attribute is determined by the corresponding configuration data.

Attribute Type: Unsigned8

Allowed values: 1 to 128

Format

This attribute defines the format of the Output Data Element. The value of this attribute is determined by the corresponding configuration data.

Allowed values:

Table 127 – Format

Value	Meaning
0	Byte
1	Word

Consistency

This attribute defines the Consistency of the Output Data Element. The value of this attribute is determined by the corresponding configuration data.

Allowed values:

Table 128 – Consistency

Value	Meaning
0	Consistency over the defined Format of the Output Data Element
1	Consistency over the whole Output Data Element

Name

This optional attribute defines the name of the Output Data Element. The value of this attribute is determined by the corresponding configuration data.

Attribute Type: Visible String(32)

9.2.2.2.2 Invocation of the Simple Output Data object

For the invocation of the Simple Output Data object the following rules have to be considered:

- Simple Output Data objects shall not exceed the total length of 244 octets. The attributes List of Output Data Elements and Length have to be set accordingly.
- Only one Simple Output Data object shall be invoked in a DP-Slave.
- A Simple Output Data object shall only be invoked if no Extended Output Data object (see 9.2.2.2.4) has been invoked.

9.2.2.2.3 Extended Input Data Class Specification

The Extended Input Data object is described by the following template:

DP ASE: I/O Data ASE
CLASS: Extended Input Data
CLASS ID: not used
PARENT CLASS: TOP

ATTRIBUTES:

1. (m) Key Attribute: Implicit
2. (m) Attribute: List of Input Data Elements
- 2.1. (m) Attribute: Slot Number
- 2.2. (m) Attribute: Selector of Data Type
- 2.2.1. (s) Attribute: Extended Input Data Description
- 2.2.2. (s) Attribute: Simple Input Data Description
- 2.2.2.1. (m) Attribute: Length
- 2.2.2.2. (m) Attribute: Format
- 2.2.2.3. (m) Attribute: Consistency
- 2.3. (o) Attribute: Name

SERVICES:

1. (m) OpsService: Set Input
2. (o) OpsService: Read Input
3. (m) OpsService: Get Input
4. (o) OpsService: New Input
5. (m) OpsService: Global Control

9.2.2.2.3.1 Attributes

Implicit

The attribute Implicit indicates that the Extended Input Data object is implicitly addressed by the services.

List of Input Data Elements

This attribute is composed of the following list elements:

Slot Number

This attribute defines to which module the Input Data Element belongs.

Attribute Type: Unsigned8

Allowed values: 1 to 244

Selector of Data Type

This attribute defines whether this module is described according to the Extended Input Data Description or according to the Simple Input Data Description. In case of an Extended Input Data Description, consistency over the whole Input Data Element is always performed and therefore the attribute consistency will be omitted.

Extended Input 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 5.2.1 is used.

Simple Input Data Description

This attribute consists of the following attributes.

Length

The attribute Length defines the number of octets of the Input Data Element. The value of this attribute is determined by the corresponding configuration data.

Attribute Type: Unsigned8

Allowed values: 1 to 128

Format

This attribute defines the format of the Input Data Element. The value of this attribute is determined by the corresponding configuration data.

Allowed values:

Table 129 – Format

Value	Meaning
0	Byte
1	Word

Consistency

This attribute defines the Consistency of the Input Data Element. The value of this attribute is determined by the corresponding configuration data.

Allowed values:

Table 130 – Consistency

Value	Meaning
0	Consistency over the defined Format of the Input Data Element
1	Consistency over the whole Input Data Element

Name

This optional attribute defines the name of the Input Data Element. The value of this attribute is determined by the corresponding configuration data.

Attribute Type: Visible String(32)

9.2.2.3.2 Invocation of the Extended Input Data object

For the invocation of the Extended Input Data object the following rules have to be considered:

- Extended Input Data objects shall not exceed the total length of 244 octets. The attributes List of Input Data Elements, Input Data Description respectively Format and Length have to be set accordingly.
- Only one Extended Input Data object shall be invoked in a DP-Slave.
- An Extended Input Data object shall only be invoked if no Simple Input Data object has been invoked.

9.2.2.2.4 Extended Output Data Class Specification

The Extended Output Data object is described by the following template:

DP ASE:	I/O Data ASE
CLASS:	Extended Output Data
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m)	Key Attribute: Implicit
2. (m)	Attribute: List of Output Data Elements
2.1. (m)	Attribute: Slot Number
2.2. (m)	Attribute: Selector of Data Type
2.2.1. (s)	Attribute: Extended Output Data Description
2.2.2. (s)	Attribute: Simple Output Data Description
2.2.2.1. (m)	Attribute: Length
2.2.2.2. (m)	Attribute: Format
2.2.2.3. (m)	Attribute: Consistency
2.3. (o)	Attribute: Name
SERVICES:	
1. (m)	OpService: Set Output
2. (o)	OpService: Read Output
3. (m)	OpService: Get Output
4. (o)	OpService: New Output
5. (m)	OpService: Global Control

9.2.2.2.4.1 Attributes

Implicit

The attribute Implicit indicates that the Extended Output Data object is implicitly addressed by the services.

List of Output Data Elements

This attribute is composed of the following list elements:

Slot Number

This attribute defines to which module the Output Data Element belongs.

Attribute Type: Unsigned8

Allowed values: 1 to 244

Selector of Data Type

This attribute defines whether this module is described according to the Extended Output Data Description or according to the Simple Output Data Description. In case of an Extended Output Data Description, consistency over the whole Output Data Element is always performed and therefore the attribute consistency will be omitted.

Extended Output 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 5.2.1 is used.

Simple Output Data Description

This attribute consists of the following attributes.

Length

The attribute Length defines the number of octets of the Output Data Element. The value of this attribute is determined by the corresponding configuration data. The value of this attribute is determined by the corresponding configuration data.

Attribute Type: Unsigned8

Allowed values: 1 to 128

Format

This attribute defines the format of the Output Data Element. The value of this attribute is determined by the corresponding configuration data. The value of this attribute is determined by the corresponding configuration data.

Allowed values:

Table 131 – Format

Value	Meaning
0	Byte
1	Word

Consistency

This attribute defines the Consistency of the Output Data Element. The value of this attribute is determined by the corresponding configuration data. The value of this attribute is determined by the corresponding configuration data.

Allowed values:

Table 132 – Consistency

Value	Meaning
0	Consistency over the defined Format of the Output Data Element
1	Consistency over the whole Output Data Element

Name

This optional attribute defines the name of the Output Data Element. The value of this attribute is determined by the corresponding configuration data. The value of this attribute is determined by the corresponding configuration data.

Attribute Type: Visible String(32)

9.2.2.2.4.2 Invocation of the Extended Output Data object

For the invocation of the Extended Output Data object the following rules have to be considered:

- Extended Output Data objects shall not exceed the total length of 244 octets. The attributes List of Output Data Elements, Output Data Description respectively Format and Length have to be set accordingly.
- Only one Extended Output Data object shall be invoked in a DP-Slave.
- An Extended Output Data object shall only be invoked if no Simple Output Data object has been invoked.

9.2.2.3 I/O Data ASE Behaviour

9.2.2.3.1 Behaviour of the Input Data object

For performance reasons, the value of the Input Data object should be transferred on change as fast as possible to the MS0 buffer for transmission. The speed and the jitter of this action is a performance parameter of a DP-Slave. This parameter determines the freeze accuracy.

For highly synchronised DP-Slaves, this action should be executed always at the same moment of time before the Freeze Command. This requires that a Freeze Command will be sent cyclically.

9.2.2.3.2 Behaviour of the Output Data object

For performance reasons, the value for the Output Data object should be transferred on receipt as fast as possible to the object. The speed and the jitter of this action is a performance parameter of a DP-Slave. This parameter determines the synchronisation accuracy.

For highly synchronised DP-Slaves, this action should be executed always at the same moment of time after the Sync Command.

9.2.2.4 I/O Data Service Specification

9.2.2.4.1 Set Input

This service shall be used by a DP-Slave to pass the value of a Simple Input Data object or an Extended Input Data object to the DP Application Layer. This service shall only be used in conjunction with the MS0 AR.

By this service the server application process of the DP-Slave updates the local buffer. According to the buffer to buffer characteristics of the used communication relationship the following behaviour is possible:

- The Set Input requests come faster than the contents of the buffers are transported over the network. In this case not each value of the Input Data object provided with the Set Input service is conveyed over the network. Only the latest value is transported, others are locally overwritten.
- The Set Input requests issued slower than the contents of the buffers are transported over the network. In this case each value of the Input Data object provided with the Set Input service is conveyed more than once over the network.
- If the Set Input requests are issued synchronous with the transport of the contents of the buffers each value of the Input Data object provided with the Set Input service is conveyed over the network.

Table 133 – Set Input

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Input 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.

Input Data

This parameter contains the value of the Input Data object.

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

9.2.2.4.2 Read Input

By this service any DP-Master (Class 2) can read the value of the Input Data object of a DP-Slave. The service will only deliver a positive result if the DP-Slave is already in the data exchange mode with its assigned DP-Master (Class 1).

Table 134 – Read Input

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Result (+)		S
AREP		M
Input Data		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.

Result (+)

This selection type parameter indicates that the service request succeeded.

Input Data

This parameter contains the requested value of the Input Data object.

Result (-)

This selection type parameter indicates that the service request failed.

Status

This parameter provides detailed information about the occurred error.

Allowed values: DS, NA, RS, UE, NR, RE

9.2.2.4.3 Get Input

This service may be used by a DP-Master (Class 1) to read:

- the values of all Input Data objects of the assigned DP-Slaves if the parameter CREP and Slot Number have not been used in the Get Input service request
- the value of the Input Data object of a single assigned DP-Slave if the parameter Slot Number has not been used in the Get Input service request
- the value of an Input Data Element of a single assigned DP-Slave if the parameter CREP and Slot Number have been used in the Get Input service request

Table 135 – Get Input

Parameter name	.req	.cnf
Argument	M	
AREP	M	
CREP	U	
Slot Number	U	
Result (+)		S
AREP		M
CREP		C
Slot Number		C
Input Data		M
Result (-)		S
AREP		M
CREP		C
Slot Number		C

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

If this parameter is present then the Input Data object of a single DP-Slave is addressed.

Slot Number

If this parameter is present then an Input Data Element of a single DP-Slave is addressed. If this parameter is used the parameter CREP shall also be present.

Allowed values: 1 to 244

Result (+)

This selection type parameter indicates that the service request succeeded.

Input Data

This parameter contains the selected value.

Result (-)

This selection type parameter indicates that the service request failed.

9.2.2.4.4 New Input

The New Input service signals to the application of the DP-Master (Class 1) that new values of all Input Data objects of all assigned DP-Slaves which are related to the MS0 AR are present.

Table 136 – New Input

Parameter name	.ind
Argument	M
AREP	M

Argument

The argument shall convey the service specific parameters of the service indication.

AREP

This parameter is the local identifier of the AR which passed the indication.

9.2.2.4.5 Set Output

By this service the client application process of the DP-Master (Class 1) requests the conveyance of:

- the values of all Output Data objects of the assigned DP-Slaves if the parameter CREP and Slot Number have not been used in the Set Output service request
- the value of the Output Data object of a single assigned DP-Slave if the parameter Slot Number has not been used in the Set Output service request
- the value of an Output Data Element of a single assigned DP-Slave if the parameter CREP and Slot Number have been used in the Set Output service request

According to the buffer to buffer characteristics the following behaviour is possible:

- The Set Output requests come 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 137 – Set Output

Parameter name	.req	.cnf
Argument	M	
AREP	M	
CREP	U	
Slot Number	U	
Output Data	M	
Result (+)		S
AREP		M
CREP		C
Slot Number		C
Result (-)		S
AREP		M
CREP		C
Slot Number		C

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

If this parameter is present then the Output Data object of a single DP-Slave is addressed.

Slot Number

If this parameter is present then an Output Data Element of a single DP-Slave is addressed. If this parameter is used the parameter CREP shall also be present.

Allowed values: 1 to 244

Output Data

This parameter contains the selected value.

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

9.2.2.4.6 Read Output

By this service any DP-Master (Class 2) can read the value of the Output Data object of a DP-Slave. The service will only deliver a positive result if the DP-Slave is already in the data exchange mode with its assigned DP-Master (Class 1).

Table 138 – Read Output

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Result (+)		S
AREP		M
Output Data		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.

Result (+)

This selection type parameter indicates that the service request succeeded.

Output Data

This parameter contains the requested value of the Output Data object.

Result (-)

This selection type parameter indicates that the service request failed.

Status

This parameter provides detailed information about the occurred error.

Allowed values: DS, NA, RS, UE, NR, RE

9.2.2.4.7 Get Output

This service shall be used by a DP-Slave to read the value for its Simple Output Data object or Extended Output Data object from the DP Application Layer. This service shall only be used in conjunction with the MS0 AR.

Table 139 – Get Output

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Result (+)		S
AREP		M
Output Data		M
Clear Flag		M
New Flag		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.

Result (+)

This selection type parameter indicates that the service request succeeded.

Output Data

This parameter contains the value for the Output Data object.

Clear Flag

This parameter indicates whether the value for the Output Data object is valid or cleared.

Allowed values:

Table 140 – Clear Flag

Value	Meaning
0	value for the Output Data object valid (DP-Master in the state OPERATE)
1	value for the Output Data cleared (DP-Master in the state CLEAR)

New Flag

This parameter indicates whether a new value for the Output Data object is received or not.

Allowed values:

Table 141 – New Flag

Value	Meaning
0	no new value received
1	new value received

Result (-)

This selection type parameter indicates that the service request failed.

9.2.2.4.8 New Output

This service indicates to the application of the DP-Slave that a new value for its Simple Output Data object or Extended Output Data object is present.

Table 142 – New Output

Parameter name	.ind
Argument	M
AREP	M
Clear Flag	M

Argument

The argument shall convey the service specific parameters of the service indication.

AREP

This parameter is the local identifier of the AR which passed the indication.

Clear Flag

This parameter indicates whether the value for the Output Data object is valid or cleared.

Allowed values:

Table 143 – Clear Flag

Value	Meaning
0	value for the Output Data object valid (DP-Master in the state OPERATE)
1	value for the Output Data cleared (DP-Master in the state CLEAR)

9.2.2.4.9 Global Control

This service is used by the DP-Master (Class 1) for the synchronisation of the values of the Input Data objects (Freeze Command) and the synchronisation of the values of the Output Data objects (Sync Command) of a group or all assigned DP-Slaves.

Furthermore this service is used by the DP-Master (Class 1) to send information about its own operation mode (Clear Command) to all assigned DP-Slaves.

Table 144 – Global Control

Parameter name	.req	.ind	.cnf
Argument	M	M(=)	
AREP	M	M(=)	
Clear Command		M	
Sync Command	M	M(=)	
Freeze Command	M	M(=)	
Group Select	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.

Clear Command

This parameter contains the Clear Command.

Allowed values:

Table 145 – Clear Command

Value	Meaning
0	DP-Master (Class 1) in the state OPERATE: The values for the Output Data objects are valid.
1	DP-Master (Class 1) in the state CLEAR: The values for the Output Data objects shall be set to a value corresponding to a safe state.

Sync Command

This parameter controls the Sync Operation.

Allowed values:

Table 146 – Sync Command

Value	Meaning
0	no action
1	Sync
2	Unsync

Freeze Command

This parameter controls the Freeze Operation.

Allowed values:

Table 147 – Freeze Command

Value	Meaning
0	no action
1	Freeze
2	Unfreeze

Group Select

This parameter determines which group(s) of assigned DP-Slaves shall be addressed. The control command takes effect if the AND operation between the attribute Group Identifier of the ARL object of the DP-Slave and the Group Select parameter results in a value different from zero. If the parameter Group Select is zero all assigned DP-Slaves are addressed.

Result (+)

This selection type parameter indicates that the service request has been transmitted.

Result (-)

This selection type parameter indicates that the service request failed.

Status

This parameter provides detailed information about the occurred error.

Allowed values: DS, NO, IV

9.2.3 Diagnosis ASE

9.2.3.1 Overview

In the DP Application Layer environment, the Diagnosis ASE of a DP-Slave contains Diagnosis objects that are used to report the status of the Default Application Process of the DP-Slave and the modules which are part of the configuration. The Diagnosis ASE defines attributes of the Diagnosis objects and provides a set of local services used to set and get their values. The following Diagnosis objects are defined:

- Device Related Diagnosis object
- Status object
- Module Status Data object
- Identifier Related Diagnosis object
- Channel Related Diagnosis object

The application of a DP-Slave may provide the value of one or several Diagnosis objects to the MS0 AR. The DP-Slave shall provide only the values of those Diagnosis objects whose values have changed since their last provision. The DP-Slave may indicate (with the parameter Ext Diag Overflow) that the values of all Diagnosis objects with a changed value could not be provided with one Set Slave Diag service request. The DP-Master (Class 1) shall indicate (with the parameter Ext Diag Overflow) in the Get Slave Diag confirmation that not all received values of the Diagnosis objects from the DP-Slave could be provided to the application.

The DP-Slave shall indicate (with the parameter Ext Diag Flag) that a relevant diagnosis event has occurred which leads to a change in at least one Diagnosis object (e. g. should be set if one element of the Identifier Related Diagnosis object is set to the value 1). This flag causes an entry in the attribute System Diagnosis of the Master Diag object of the DP-Master (Class 1).

The Diagnosis model uses the client/server access model. The Diagnosis model is based on a buffer to buffer relationship. Thus the server application updates (sets) the values of the Diagnosis objects in its local buffer for the conveyance. The client application may get the values of the Diagnosis objects from its local buffer.

The DP-Master (Class 1) communicates via the MS0 AR with all Diagnosis objects of the assigned DP-Slaves simultaneously. Additionally the client application may communicate with the Diagnosis objects of a single DP-Slave.

The DP-Master (Class 2) communicates via the MS0 AR with the Diagnosis objects of a single DP-Slave.

Furthermore additional Diagnosis information (attributes of the MS0 AR object) is provided by the DP Application Layer as parameters of the Get Slave Diag service and Read Slave Diag service.

The formal description of the various Diagnosis objects is presented next, followed by a description of its services.

9.2.3.2 Diagnosis Class Specification

9.2.3.2.1 Device Related Diagnosis Class Specification

The Device Related Diagnosis object is described by the following template:

DP ASE:	Diagnosis ASE
CLASS:	Device Related Diagnosis
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m) Key Attribute:	Implicit
2. (m) Attribute:	Device Related Data Description
SERVICES:	
1. (m) OpsService:	Set Slave Diag
2. (m) OpsService:	Get Slave Diag
3. (o) OpsService:	Read Slave Diag
4. (o) OpsService:	New Slave Diag

9.2.3.2.1.1 Attributes

Implicit

The attribute Implicit indicates that the Device Related Diagnosis object is implicitly addressed by the services.

Device Related 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 5.2.1 is used. The maximum length of this Data Description shall not exceed 62 octets.

9.2.3.2.1.2 Invocation of the Device Related Diagnosis object

For the invocation of the Device Related Diagnosis object the following rules shall be considered:

- Device Related Diagnosis objects shall be used only if the device attribute DPV1 = FALSE.
- Only one Device Related Diagnosis object shall be invoked in a DP-Slave.

9.2.3.2.2 Identifier Related Diagnosis Class Specification

This class specifies the state of the modules which are part of the configuration. Each bit is related to a Module (Slot). An element set to FALSE means that no diagnosis of the related module exists or no module is placed. A bit set to TRUE indicates a module failure. If a module failure exists, the values of the Input Data object and/or the Output Data object of this module are set to the default value by the DP-Slave.

The Identifier Related Diagnosis object is described by the following template:

DP ASE: **Diagnosis ASE**
CLASS: **Identifier Related Diagnosis**
CLASS ID: not used
PARENT CLASS: TOP
ATTRIBUTES:
 1. (m) Key Attribute: Implicit
 2. (m) Attribute: List of Identifier Status
 2.1. (m) Attribute: Identifier Status
SERVICES:
 1. (m) OpService: Set Slave Diag
 2. (m) OpService: Get Slave Diag
 3. (o) OpService: Read Slave Diag
 4. (o) OpService: New Slave Diag

9.2.3.2.2.1 Attributes

Implicit

The attribute Implicit indicates that the Identifier Related Diagnosis object is implicitly addressed by the services.

List of Identifier Status

This attribute is composed of the following list elements:

Identifier Status

The elements of this list attribute indicates the state of each module. E.g. if the first module has an error the first element of the List of Identifier Status has the value 1.

Allowed values:

Table 148 – Identifier Status

Value	Meaning
0	Module okay.
1	Module not okay.

9.2.3.2.2.2 Invocation of the Identifier Related Diagnosis object

For the invocation of the Identifier Related Diagnosis object the following rules shall be considered:

- Only one Identifier Related Diagnosis object shall be invoked in a DP-Slave.

9.2.3.2.3 Channel Related Diagnosis Class Specification

The Channel Related Diagnosis object is described by the following template:

DP ASE:	Diagnosis ASE
CLASS:	Channel Related Diagnosis
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m) Key Attributes:	Identifier
2. (m) Attribute:	Channel Type
3. (m) Attribute:	IO Type
4. (m) Attribute:	Error Type
SERVICES:	
1. (m) OpsService:	Set Slave Diag
2. (m) OpsService:	Get Slave Diag
3. (o) OpsService:	Read Slave Diag
4. (o) OpsService:	New Slave Diag

9.2.3.2.3.1 Attributes

Identifier

This key attribute is composed of Identifier Number and Channel Number to indicate the source of the Channel Related Diagnosis object.

Allowed values: 0 to 63, 0 to 63

Channel Type

This attribute indicates the channel-size to which the Channel Related Diagnosis object is related.

Allowed values:

Table 149 – Channel Type

Value	Meaning
0	unspecific
1	1 bit
2	2 bit
3	4 bit
4	byte
5	word
6	2 words

IO Type

This attribute indicates the IO Type of the channel to which the Channel Related Diagnosis object is related.

Allowed values:

Table 150 – IO Type

Value	Meaning
1	an Input channel
2	an Output channel
3	a Bi-directional channel

Error Type

This attribute indicates the error type of the Channel Related Diagnosis object.

Allowed value:

Table 151 – Error Type

Value	Meaning
1	short circuit
2	undervoltage
3	overvoltage
4	overload
5	overtemperature
6	line break
7	upper limit value exceeded
8	lower limit value exceeded
9	error
16 to 31	manufacturer specific

9.2.3.2.3.2 Invocation of the Channel Related Diagnosis object

For the invocation of the Channel Related Diagnosis object the following rules shall be considered:

- Several Channel Related Diagnosis objects may be invoked in a DP-Slave.

9.2.3.2.4 Status Class Specification

The status model allows the transfer of status messages from the DP-Slave to the DP-Master (Class 1). The status model is slot based.

A Status object is described by the following template:

DP ASE: **Diagnosis ASE**
CLASS: **Status**
CLASS ID: not used
PARENT CLASS: TOP

ATTRIBUTES:

1. (m) Key Attributes: Identifier
2. (m) Attribute: Status Specifier
3. (o) Attribute: Status Data Description

SERVICES:

1. (m) OpsService: Set Slave Diag
2. (m) OpsService: Get Slave Diag
3. (o) OpsService: Read Slave Diag
4. (o) OpsService: New Slave Diag

9.2.3.2.4.1 Attributes**Identifier**

This key attribute is composed of Slot Number and Status Type to indicate the number of the slot and the status type. The following status types are possible:

Allowed values for Slot Number: 0 to 254

Allowed values for Status Type:

Table 152 – Status Type

Value	Meaning
1	Status Message
32-126	Manufacturer Specific

Status Specifier

This attribute specifies additional status information.

Allowed values:

Table 153 – Status Specifier

Value	Meaning
0	no further differentiation
1	status appears
2	status disappears

Status 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 5.2.1 is used.

9.2.3.2.4.2 Invocation of the Status object

For the invocation of the Status object the following rules have to be considered:

- #### 9.2.3.2.5 Module Status Data Class Specification

The Module Status Data object is described by the following template:

SERVICES:

1. (m) OpsService: Set Slave Diag
2. (m) OpsService: Get Slave Diag
3. (o) OpsService: Read Slave Diag
4. (o) OpsService: New Slave Diag

Implicit

Status Specifier

Allowed values:

Table 154 – Status Specifier

Value	Meaning
0	no further differentiation
1	status appears
2	status disappears

This attribute is composed of the following list elements:

This attribute specifies additional status information for each module. The first element of the list contains the status of the first module (Slot Number 1).

Allowed values:

Table 155 – Module Status

Value	Meaning
0	data valid
1	data invalid - signal error: the data of the corresponding module are not valid due to an error (inputs not accessible or outputs cannot be set, e.g. due to short circuit or overload)
2	data invalid - wrong module: the data of the corresponding module are not valid, due to a wrong module in place
3	data invalid - no module: the data of the corresponding module are not valid because there is no module in place

9.2.3.2.5.2 Invocation of the Module Status Data object

For the invocation of the Module Status Data object the following rules have to be considered:

- The Module Status Data object shall be used only if the device attribute DPV1 = TRUE.
- Only one Module Status Data object shall be invoked in a DP-Slave.
- The Module Status Data object shall not exceed the total length of 59 octets.

9.2.3.3 Diagnosis Service Specification

9.2.3.3.1 Set Slave Diag

This service shall be used by a DP-Slave to pass the values of the Diagnosis objects to the DP Application Layer. The overall transmitted diagnosis information shall not exceed the total length of 238 octets. By this service the server application process of the DP-Slave updates the local buffer. According to the buffer to buffer characteristics of the MS0 AR the following behaviour is possible:

- The Set Slave Diag requests come faster than the contents of the buffers are transported over the network. In this case not each value of the Diagnosis objects provided with the Set Slave Diag service is conveyed over the network. Only the latest values are transported, others are locally overwritten.
- The Set Slave Diag requests issued slower than the contents of the buffers are transported over the network. In this case the values of the Diagnosis objects provided with the Set Slave Diag service are conveyed more than once over the network.
- If the Set Slave Diag requests are issued synchronous with the transport of the contents of the buffers each value of the Diagnosis objects provided with the Set Slave Diag service are conveyed over the network.

Table 156 – Set Slave Diag

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Ext Diag Overflow	M	
Ext Diag Flag	M	
Ext Diag Data	C	
Device Related Diagnosis Data	U	
List of Identifier Status	U	
Identifier Status	M	
List of Channel Related Diagnosis Data	U	
Identifier Number	M	
Channel Number	M	
Channel Type	M	
IO Type	M	
Error Type	M	
List of Status Data	U	
Slot Number	M	
Status Type	M	
Status Specifier	M	
Status Data	M	
Module Status Data	U	
Status Specifier	M	
List of Module Status	M	
Module Status	M	
Revision Number	U	
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.

Ext Diag Flag

This parameter indicates that a relevant diagnosis event has occurred which leads to a change in at least one Diagnosis object (e. g. should be set if one element of the Identifier Related Diagnosis object is set to the value 1).

Allowed values:

Table 157 – Ext Diag Flag

Value	Meaning
TRUE	There exists at least one important diagnosis (such as short circuit). An entry has to be present in the Diagnosis objects (Device Related Diagnosis objects, Identifier Related Diagnosis object, Channel Related Diagnosis objects, Status objects, Module Status object).
FALSE	No important diagnosis entry exist. If the attribute value is FALSE, a status message (for example, supply voltage OK) may be present in the Diagnosis Data, or there exists no Diagnosis Data. The meaning of this status message is application specific.

Ext Diag Overflow

This parameter indicates that the values of all Diagnosis objects with a changed value could not been provided with this Set Slave Diag service request.

Allowed values: TRUE, FALSE

NOTE In case of an overflow condition it is allowed to truncate only at Diagnosis object limits.

Ext Diag Data

This parameter is composed of the following items.

Device Related Diagnosis Data

This parameter contains the value of the Device Related Diagnosis object (this parameter shall be used only if DPV1 = FALSE).

List of Identifier Status

This parameter is composed of the following list elements:

Identifier Status

This parameter contains the value of the attribute Identifier Status of the Identifier Related Diagnosis object.

List of Channel Related Diagnosis Data

This parameter is composed of the following list elements:

Identifier Number

This parameter contains the value of the attribute Identifier Number of one Channel Related Diagnosis object.

Channel Number

This parameter contains the value of the attribute Channel Number of one Channel Related Diagnosis object.

Channel Type

This parameter contains the value of the attribute Channel Type of one Channel Related Diagnosis object.

IO Type

This parameter contains the value of the attribute IO Type of one Channel Related Diagnosis object.

Error Type

This parameter contains the value of one Channel Related Diagnosis object.

List of Status Data

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the attribute Slot Number of one Status object.

Status Type

This parameter contains the value of the attribute Status Type for one Status object.

Status Specifier

This parameter contains the value of the attribute Status Specifier of one Status object.

Status Data

This parameter contains the value of one Status object.

Module Status Data

This parameter is composed of the following subparameters:

Status Specifier

This parameter contains the value of the attribute Status Specifier of the Module Status Data object.

List of Module Status

This parameter is composed of the following list elements:

Module Status

This parameter contains the value of the attribute Module Status of the Module Status Data object.

Revision Number

This parameter contains the actual value of the Revision Number.

Result

This parameter indicates that the service request succeeded.

9.2.3.3.2 Get Slave Diag

This service may be used by a DP-Master (Class 1) to read the previously received values of the Diagnosis objects of one or all assigned DP-Slave which are related to the MS0 AR. Additionally the Get Slave Diag service provides the diagnosis information provided by the DP Application Layer of the DP-Slave and the DP-Master (Class 1).

Table 158 – Get Slave Diag

Parameter name	.req	.cnf
Argument	M	
AREP	M	
CREP	U	
Result (+)		S
AREP		M
CREP		C
List of Diag Data		M
Prm Req		M
Prm Fault		M
Cfg Fault		M
Not Supported		M
Station Not Ready		M
Stat Diag		M
WD On		M
Sync Mode		M
Freeze Mode		M
Master Add		M
Ident Number		M
Deactivated		M
Station Not Existent		M
Invalid Slave Response		M
Master Lock		M
Ext Diag Overflow		M
Ext Diag Flag		M
Ext Diag Data		C
Device Related Diagnosis Data		U
List of Identifier Status		U
Identifier Status		M
List of Channel Related Diagnosis Data		U
Identifier Number		M
Channel Number		M
Channel Type		M
IO Type		M
Error Type		M
List of Status Data		U
Slot Number		M
Status Type		M
Status Specifier		M

(continued on next page)

Status Data		M
Module Status Data		U
Status Specifier		M
List of Module Status		M
Module Status		M
Revision Number		U
Result (-)		S
AREP		M
CREP		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

If this parameter is present then the Diagnosis objects of a single DP-Slave are addressed.

Result (+)

This selection type parameter indicates that the service request succeeded.

List of Diag Data

This parameter is composed of the following list elements:

Prm Req

This parameter contains the value of the attribute Prm Req of the MS0 ARL entry of a DP-Slave.

Prm Fault

This parameter contains the value of the attribute Prm Fault of the MS0 ARL entry of a DP-Slave.

Cfg Fault

This parameter contains the value of the attribute Cfg Fault of the MS0 ARL entry of a DP-Slave.

Not Supported

This parameter contains the value of the attribute Not Supported of the MS0 ARL entry of a DP-Slave.

Station Not Ready

This parameter contains the value of the attribute Station Not Ready of the MS0 ARL entry of a DP-Slave.

Stat Diag

This parameter contains the value of the attribute Stat Diag of the MS0 ARL entry of a DP-Slave.

WD On

This parameter contains the value of the attribute WD On of the MS0 ARL entry of a DP-Slave.

Sync Mode

This parameter contains the value of the attribute Sync Mode of the MS0 ARL entry of a DP-Slave.

Freeze Mode

This parameter contains the value of the attribute Freeze Mode of the MS0 ARL entry of a DP-Slave.

Master Add

This parameter contains the value of the attribute Master Add of the MS0 ARL entry of a DP-Slave.

Ident Number

This parameter contains the value of the attribute Ident Number of the MS0 ARL entry of a DP-Slave.

Deactivated

This parameter contains the value of the attribute Deactivated of the MS0 ARL entry of the DP-Master.

Station Non Existent

This parameter contains the value of the attribute Station Non Existent of the MS0 ARL entry of the DP-Master.

Invalid Slave Response

This parameter contains the value of the attribute Invalid Slave Response of the MS0 ARL entry of the DP-Master.

Master Lock

This parameter indicates whether the DP-Slave is locked by a DP-Master. If the Master Lock parameter is TRUE the DP-Slave is locked by another DP-Master. If the parameter is False the DP-Slave is not locked by another DP-Master.

Ext Diag Overflow

This parameter contains the value of the parameter Ext Diag Overflow of the Set Slave Diag service if all delivered values of the Diagnosis objects are contained in this service confirmation. This parameter contains the value of the attribute Ext Diag Overflow of the MS0 ARL entry of the DP-Master if not all delivered values of the Diagnosis objects are contained in this service confirmation due to storage problems.

Allowed values: TRUE, FALSE

NOTE In case of an overflow condition it is allowed to truncate only at Diagnosis object limits.

Ext Diag Flag

This parameter contains the value of the parameter Ext Diag Flag of the Set Slave Diag service of a DP-Slave.

Ext Diag Data

This parameter is composed of the following subparameters:

Device Related Diagnosis Data

This parameter contains the value of the Device Related Diagnosis object of a DP-Slave.

List of Identifier Status

This parameter is composed of the following list elements:

Identifier Status

This parameter contains the value of the attribute Identifier Status of the Identifier Related Diagnosis object of a DP-Slave.

List of Channel Related Diagnosis Data

This parameter is composed of the following list elements:

Identifier Number

This parameter contains the value of the attribute Identifier Number of one Channel Related Diagnosis object of a DP-Slave.

Channel Number

This parameter contains the value of the attribute Channel Number of one Channel Related Diagnosis object of a DP-Slave.

Channel Type

This parameter contains the value of the attribute Channel Type of one Channel Related Diagnosis object of a DP-Slave.

IO Type

This parameter contains the value of the attribute IO Type of one Channel Related Diagnosis object of a DP-Slave.

Error Type

This parameter contains the value of one Channel Related Diagnosis object of a DP-Slave.

List of Status Data

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the attribute Slot Number of one Status object of a DP-Slave.

Status Type

This parameter contains the value of the attribute Status Type for one Status object of a DP-Slave.

Status Specifier

This parameter contains the value of the attribute Status Specifier of one Status object of a DP-Slave.

Status Data

This parameter contains the value of one Status object of a DP-Slave.

Module Status Data

This parameter is composed of the following subparameters:

Status Specifier

This parameter contains the value of the attribute Status Specifier of the Module Status Data object of a DP-Slave.

List of Module Status

This parameter is composed of the following list elements:

Module Status

This parameter contains the value of the attribute Module Status of the Module Status Data object of a DP-Slave.

Revision Number

This parameter contains the actual value of the Revision Number.

Result (-)

This selection type parameter indicates that the service request failed.

9.2.3.3.3 Read Slave Diag

This service may be used by a DP-Master (Class 2) to read the values of the Diagnosis objects of one DP-Slave via a MS0 AR. Additionally the Read Slave Diag service provides the diagnosis information provided by the DP Application Layer of the DP-Slave and the DP-Master (Class 2).

Table 159 – Read Slave Diag

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Result (+)		S
AREP		M
Diag Data		M
Prm Req		M
Prm Fault		M
Cfg Fault		M
Not Supported		M
Station Not Ready		M
Stat Diag		M
WD On		M
Sync Mode		M
Freeze Mode		M
Master Add		M
Ident Number		M
Station Not Existent		M
Invalid Slave Response		M
Master Lock		M
Ext Diag Overflow		M
Ext Diag Flag		M
Ext Diag Data		C
Device Related Diagnosis Data		U
List of Identifier Status		U
Identifier Status		M
List of Channel Related Diagnosis Data		U
Identifier Number		M
Channel Number		M
Channel Type		M
IO Type		M
Error Type		M
List of Status Data		U
Slot Number		M
Status Type		M
Status Specifier		M
Status Data		M

(continued on next page)

Module Status Data		U
Status Specifier		M
List of Module Status		M
Module Status		M
Revision Number		U
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.

Result (+)

This selection type parameter indicates that the service request succeeded.

Diag Data

This parameter is composed of the following list elements:

Prm Req

This parameter contains the value of the attribute Prm Req of the MS0 ARL entry of the selected DP-Slave.

Prm Fault

This parameter contains the value of the attribute Prm Fault of the MS0 ARL entry of the selected DP-Slave.

Cfg Fault

This parameter contains the value of the attribute Cfg Fault of the MS0 ARL entry of the selected DP-Slave.

Not Supported

This parameter contains the value of the attribute Not Supported of the MS0 ARL entry of the selected DP-Slave.

Station Not Ready

This parameter contains the value of the attribute Station Not Ready of the MS0 ARL entry of the selected DP-Slave.

Stat Diag

This parameter contains the value of the attribute Stat Diag of the MS0 ARL entry of the selected DP-Slave.

WD On

This parameter contains the value of the attribute WD On of the MS0 ARL entry of the selected DP-Slave.

Sync Mode

This parameter contains the value of the attribute Sync Mode of the MS0 ARL entry of the selected DP-Slave.

Freeze Mode

This parameter contains the value of the attribute Freeze Mode of the MS0 ARL entry of the selected DP-Slave.

Master Add

This parameter contains the value of the attribute Master Add of the MS0 ARL entry of the selected DP-Slave.

Ident Number

This parameter contains the value of the attribute Ident Number of the MS0 ARL entry of the selected DP-Slave.

Station Non Existent

This parameter contains the value of the attribute Station Non-Existent of the MS0 ARL entry of the DP-Master.

Invalid Slave Response

This parameter contains the value of the attribute Invalid Slave Response of the MS0 ARL entry of the DP-Master.

Master Lock

This parameter indicates whether the DP-Slave is locked by a DP-Master. If the Master Lock parameter is TRUE the DP-Slave is locked by another DP-Master. If the parameter is False the DP-Slave is not locked by another DP-Master.

Ext Diag Overflow

This parameter contains the value of the parameter Ext Diag Overflow of the Set Slave Diag service if all delivered values of the Diagnosis objects are contained in this service confirmation. This parameter contains the value of the attribute Ext Diag Overflow of the MS0 ARL entry of the DP-Master if not all delivered values of the Diagnosis objects are contained in this service confirmation due to storage problems.

Allowed values: TRUE, FALSE

NOTE In case of an overflow condition it is allowed to truncate only at Diagnosis object limits.

Ext Diag Flag

This parameter contains the value of the parameter Ext Diag Flag of the Set Slave Diag service of the selected DP-Slave.

Ext Diag Data

This parameter is composed of the following subparameters:

Device Related Diagnosis Data

This parameter contains the value of the Device Related Diagnosis object of the selected DP-Slave.

List of Identifier Status

This parameter is composed of the following list elements:

Identifier Status

This parameter contains the value of the attribute Identifier Status of the Identifier Related Diagnosis object of the selected DP-Slave.

List of Channel Related Diagnosis Data

This parameter is composed of the following list elements:

Identifier Number

This parameter contains the value of the attribute Identifier Number of one Channel Related Diagnosis object of the selected DP-Slave.

Channel Number

This parameter contains the value of the attribute Channel Number of one Channel Related Diagnosis object of the selected DP-Slave.

Channel Type

This parameter contains the value of the attribute Channel Type of one Channel Related Diagnosis object of the selected DP-Slave.

IO Type

This parameter contains the value of the attribute IO Type of one Channel Related Diagnosis object of the selected DP-Slave.

Error Type

This parameter contains the value of one Channel Related Diagnosis object of the selected DP-Slave.

List of Status Data

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the attribute Slot Number of one Status object of the selected DP-Slave.

Status Type

This parameter contains the value of the attribute Status Type for one Status object of the selected DP-Slave.

Status Specifier

This parameter contains the value of the attribute Status Specifier of one Status object of the selected DP-Slave.

Status Data

This parameter contains the value of one Status object of the selected DP-Slave.

Module Status Data

This parameter is composed of the following subparameters:

Status Specifier

This parameter contains the value of the attribute Status Specifier of the Module Status Data object of the selected DP-Slave.

List of Module Status

This parameter is composed of the following list elements:

Module Status

This parameter contains the value of the attribute Module Status of the Module Status Data object of the selected DP-Slave.

Revision Number

This parameter contains the value of the Revision Number of the selected DP-Slave.

Result (-)

This selection type parameter indicates that the service request failed.

Status

This This parameter contains the reason for the failure.

Allowed values: RE, DS, UE, RS, NA, NR, RR

9.2.3.3.4 New Slave Diag

This service may be used by a DP-Master (Class 1) to indicate to the application that new received values of the Diagnosis objects of one assigned DP-Slave which is related to the MS0 AR, is present. Additionally this service may be used to indicate that new diagnosis information provided by the DP Application Layer of the DP-Slave or the DP-Master (Class 1) is present.

Table 160 – New Slave Diag

Parameter name	.ind
Argument	M
AREP	M
CREP	M

Argument

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

AREP

This parameter is the local identifier of the related AR.

CREP

This parameter is the local identifier of the related CR (DP-Slave).

9.2.4 Alarm ASE

9.2.4.1 Overview

The alarm model allows the transfer of an alarm from the DP-Slave to the assigned DP-Master (Class 1) and the explicit acknowledgement of the alarm by the DP-Master (Class 1) via the MS1 AR.

Figure 50 shows the alarm interactions between the DP-Slave (Alarm Notification) and the DP-Master (Alarm Ack).

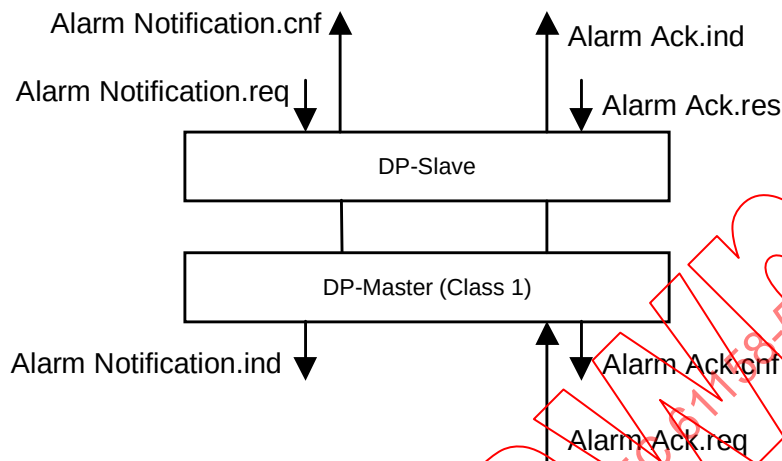


Figure 50 – Treatment of an alarm in the DP-System

For the transmission of an alarm the following conditions have to be fulfilled:

- the DP-Slave has to be in the data exchange mode
- the MS1 AR has to be open
- the corresponding alarm type has to be enabled
- the limit of active alarms is not exceeded

In addition to the alarm transmission a Channel Related and/or Identifier Related Diagnosis may be issued which identify the exact source of the alarm.

Two alternatives of the alarm handling are defined:

- only one alarm of a specific alarm type can be active at the same time
- several alarms (2 to 32) independent of the type can be active at the same time

Alarms will be queued. The alarm remains active until the DP-Master (Class 1) explicitly acknowledges the alarm.

The alarm acknowledgement is either performed via the acyclic read and write CR or optionally via a separate CR on the MS1 AR. With this latter feature, the alarms will not be delayed by read and write services.

The following alarm types are defined. They may be extended specific to the manufacturer:

Diagnosis Alarm

A diagnosis alarm signals an event within a module, for instance overtemperature, short circuit, etc.

Process Alarm

A process alarm signals the occurrence of an event in the connected process, for instance upper limit value exceeded.

Pull Alarm

A slot signals the withdrawal of a module.

Plug Alarm

A slot signals the insertion of a module.

Status Alarm

A status alarm signals a change in the state of a module, for instance run, stop or ready.

Update Alarm

An update alarm signals the change of a parameter in a module e.g. by a local operation or remote access.

9.2.4.2 Alarm Class Specification

An Alarm object is described by the following template:

DP ASE:	Alarm ASE
CLASS:	ALARM DATA
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m) Key Attributes:	Identifier
2. (m) Attribute:	Add Ack
3. (m) Attribute:	Alarm Specifier
4. (o) Attribute:	Alarm Data Description
SERVICES:	
1. (m) OpService:	Alarm Notification
2. (m) OpService:	Alarm Ack

9.2.4.2.1 Attributes

Identifier

This key attribute is composed of Slot Number and Alarm Type to indicate the number of the slot and the alarm type.

Allowed values for Slot Number: 0 to 254

Allowed values for Alarm Type:

Table 161 – Alarm Type

Value	Meaning
1	Diagnosis Alarm
2	Process Alarm
3	Pull Alarm
4	Plug Alarm
5	Status Alarm
6	Update Alarm
32-126	Manufacturer Specific

Add Ack

This attribute indicates to the DP-Master (Class 1) that this alarm requires an additional User acknowledgement. This can be done for instance by a write service. The alarm remains active after the Alarm Ack.

Allowed values:

Table 162 – Add Ack

Value	Meaning
TRUE	additional acknowledgement required
FALSE	no additional acknowledgement required

Alarm Specifier

This attribute specifies additional alarm information.

Allowed values:

Table 163 – Alarm Specifier

Value	Meaning
0	no further differentiation
1	Alarm appears and the related module is disturbed
2	Alarm disappears and the related module has no further errors
3	Alarm disappears and the related module is still disturbed

Alarm 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 5.2.1 is used.

9.2.4.2.2 Invocation of the Alarm Data object

For the invocation of the Alarm Data object the following rules have to be considered:

- Alarm Data objects shall be used only if the device attribute DPV1 = TRUE
- Several Alarm Data objects can be invoked in a DP-Slave
- For each module only one Alarm Data object of one type shall be invoked

9.2.4.3 Service Specification

9.2.4.3.1 Alarm Notification

This service is used to transfer an alarm notification from the DP-Slave to the DP-Master (Class 1).

Table 164 – Alarm Notification

Parameter name	.req	.cnf	.ind
Argument	M		M(=)
AREP	M		M(=)
Slot Number	M		M(=)
Alarm Type	M		M(=)
Seq Nr	M		M(=)
Add Ack	M		M(=)
Alarm Specifier	M		M(=)
Alarm Data	U		C(=)
Result (+)		S	
AREP		M	
Slot Number		M	
Alarm Type		M	
Seq Nr		M	
Result (-)		S	
AREP		M	
Slot Number		M	
Alarm Type		M	
Seq Nr		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.

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.

Seq Nr

This parameter is used as an unique identifier for the transaction of an alarm notification by the DP Application Layer.

Allowed values: 0 to 31

Add Ack

This parameter contains the value of the attribute Add Ack of the Alarm Data object.

Alarm Specifier

This parameter contains the value of the attribute Alarm Specifier of the Alarm Data object.

Alarm Data

This parameter contains the value of the Alarm Data object.

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

Status

This parameter contains the reason for the failure.

Allowed values: MSAL1S_Stopped, Not_Enabled, Limit_Expired, Sequence_Nr_Pending, No_Start

9.2.4.3.2 Alarm Ack

This service is used by the DP-Master (Class 1) to acknowledge the reception of an alarm notification which has been previously received from the DP-Slave.

Table 165 – Alarm Ack

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Slot Number	M	M(=)		
Alarm Type	M	M(=)		
Seq Nr	M	M(=)		
Result (+)			S	S
AREP			M	M(=)
Slot Number			M	M(=)
Alarm Type			M	M(=)
Seq Nr			M	M(=)
Result (-)			S	S
AREP			M	M(=)
Slot Number			M	M(=)
Alarm Type			M	M(=)
Seq Nr			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.

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.

Seq Nr

This parameter contains the sequence number of the previously received alarm notification.

Allowed values: 0 to 31

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

9.2.5 Context ASE

9.2.5.1 Overview

The Context ASE provides a set of services to parameterise and configure the Default AP of a DP-Slave itself and its modules containing I/O data. The MS0 Configuration Elements object describes the module structure, the range of input and output data and contains information about the data consistency. The Context ASE defines a set of services used to set and get their values. The application of the DP-Slave will compare the configuration data send by the DP-Master (Class 1) with the local configuration of the DP-Slave. If they match the MS0 AR to this DP-Slave will be established. The real configuration of the DP-Slave may be read by a DP-Master (Class 2) with the related service.

The Context ASE also provides the means to change the address of a DP-Slave via the MS0 AR. This may be done by a DP-Master (Class 2).

The Context ASE is used to establish or release an association between an individual AP of a DP-Master (Class 2) and an individual AP of the DP-Slave for acyclic communication. This association exclusively belongs to the Process Data ASE.

The formal description of the Context ASE is presented next, followed by a description of its services.

9.2.5.2 Context Class Specification

9.2.5.2.1 MS0 User Parameter Class Specification

The MS0 User Parameter object is described by the following template:

DP ASE:	Context ASE
CLASS:	MS0 User Parameter
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m) Key Attribute:	Implicit
2. (o) Attribute:	User Prm Data
2.1. (o) Attribute:	Device Related User Parameter
2.1.1 (m) Attribute:	Data Description
2.2. (o) Attribute:	List of Module Related User Parameter
2.2.1 (m) Attribute:	Data Description
SERVICES:	
1. (m) OpService:	Check User Prm
2. (m) OpService:	Check User Prm Result

9.2.5.2.1.1 Attributes

Implicit

The attribute Implicit indicates that the User Parameter object is implicitly addressed by the services.

User Prm Data

This attribute is an aggregate of the Device and/or List of Module Related User Parameter and is composed of the following elements.

Device Related User Parameter

This attribute is composed of following elements:

Data Description

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 5.2.1 is used.

List of Module Related User Parameter

This attribute is composed of following list elements (each element corresponds to a module).

Data Description

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 5.2.1 is used.

9.2.5.2.1.2 Invocation of the MS0 User Parameter object

For the invocation of the MS0 User Parameter object the following rules have to be considered:

- The MS0 User Parameter object shall not exceed the total length of 237 octets (only 234 if the device attribute DPV1 = TRUE).
- Only one MS0 User Parameter object can be invoked in a DP-Slave.

9.2.5.2.2 MS0 Configuration Elements Class Specification

The MS0 Configuration Elements object is described by the following template:

DP ASE:	Context ASE
CLASS:	MS0 Configuration Elements
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m) Key Attribute:	Implicit
2. (m) Attribute:	List of Cfg Data
2.1. (o) Attribute:	Input Data Reference
2.2. (o) Attribute:	Output Data Reference
2.3. (o) Attribute:	Manufacturer Specific Data
2.4. (o) Attribute:	Module Name
SERVICES:	
1. (m) OpService:	Set Cfg
2. (m) OpService:	Get Cfg
3. (m) OpService:	Check Cfg
4. (m) OpService:	Check Cfg Result

9.2.5.2.2.1 Attributes**Implicit**

The attribute Implicit indicates that the MS0 Configuration Elements object is implicitly addressed by the services.

List of Cfg Data

This attribute is an aggregate of the configuration data of a DP-Slave which is described by Cfg Identifier or Special Cfg Identifier. It shall contain at least one element. An element of the Cfg Data consists of the following attributes.

Input Data Reference

Reference to a Simple or Extended Input Data object of an Input Data Element.

Output Data Reference

Reference to a Simple or Extended Output Data object of an Output Data Element.

Manufacturer Specific Data

This attribute contains a manufacturer specific addition to the Input and/or Output Data Reference attributes. The contents are device specific or module specific.

Attribute Type: Octet string

Module Name

This attribute defines a name of a list element.

Attribute Type: Visible String(32)

9.2.5.2.2.2 Invocation of the MS0 Configuration Elements object

For the invocation of the MS0 Configuration Elements object the following rules have to be considered:

- Only one MS0 Configuration Elements object shall be invoked in a DP-Slave
- The Cfg Data shall not exceed 244 octets.

9.2.5.2.3 Remanent Parameter Class Specification

A Remanent Parameter object is described by the following template:

DP ASE:	Context ASE
CLASS:	Remanent Parameter
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	

1. (m) Key Attribute: Implicit
2. (m) Attribute: No Add Change
3. (m) Attribute: New Slave Add
4. (o) Attribute: Rem Para
- 4.1 (m) Attribute: Data Description

SERVICES:

1. (m) OpsService: Set Slave Add

9.2.5.2.3.1 Attributes

Implicit

The attribute Implicit indicates that an Remanent Parameter object is implicitly addressed by the service.

No Add Change

This attribute specifies whether it is allowed to change the DP-Slave address again. To change this attribute itself an initial reset of the DP-Slave is necessary. After the initial reset the DP-Slave takes the default address 126 and the attribute No Add Change will be set to FALSE.

Allowed values: FALSE , TRUE

New Slave Add

This attribute contains the address of the DP-Slave which is to be set.

Rem Para

This attribute contains parameters to be stored permanently.

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 5.2.1 is used.

9.2.5.2.3.2 Invocation of Rem Para object

For the invocation of the Remanent Parameter object the following rules have to be considered:

- Only one Remanent Parameter object shall be invoked in a DP-Slave.
- The Remanent Parameter object shall not exceed 242 octets.

9.2.5.2.4 MS2 User Parameter Class Specification

The MS2 User Parameter object is described by the following template:

DP ASE:	Context ASE
CLASS:	MS2 User Parameter
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m) Key Attribute:	Implicit
2. (m) Attribute:	Features Supported
2.1. (m) Attribute:	Supported Feature
3. (m) Attribute:	Profile Features Supported
3.1. (m) Attribute:	Supported Profile Feature
4. (m) Attribute:	Profile Ident Number
SERVICES:	
1. (m) OpService:	Initiate
2. (m) OpService:	Abort

9.2.5.2.4.1 Attributes

Implicit

The attribute Implicit indicates that the MS2 User Parameter object is implicitly addressed by the services.

Features Supported

This attribute is composed of the following list elements:

Supported Feature

This attribute reflects the actual features on the MS2 AR. The DP-Master (Class 2) and the DP-Slave inform each other about the supported service functionality. The DP-Slave and the DP-Master (Class 2) has the possibility to adjust its functionality according to the requirements. The User application of the DP-Master has to decide if the reported functionality is sufficient or not.

Allowed values:

Table 166 – Supported Feature

Value	Meaning
1	Read and Write service supported
2 to 16	reserved for further features

Profile Features Supported

This attribute is composed of the following list elements:

Supported Profile Feature

This attribute reflects the actual profile features on the MS2 AR. The DP-Master (Class 2) and the DP-Slave inform each other about the supported profile features regarding the used profile definition, the profile is identified by the Profile Ident Number. The User application of the DP-Master has to decide if the reported functionality is sufficient or not.

Table 167 – Supported Profile Feature

Value	Meaning
1 to 16	feature to be defined by profiles

Profile Ident Number

The attribute Profile Ident Number contains the unique identifier of a profile definition. All devices using the same profile definition have to use the same Profile Ident Number. The Profile Ident Number will be taken from the pool of Ident Numbers for vendor specific or authorised profiles. The value 0 indicates that no profile is supported.

9.2.5.3 Context Service Specification

9.2.5.3.1 Check User Prm

The DP Application Layer of the DP-Slave notifies the application when a valid parameterisation is received from the DP-Master because this service requests a validation of the User Prm Data. A further Check User Prm indication is only issued after the Check User Prm service is completed with a Check User Prm Result service.

Table 168 – Check User Prm

Parameter name	.ind
Argument	M
AREP	M
User Prm Data	U

Argument

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

AREP

This parameter is used as local identifier of the AR.

User Prm Data

This parameter contains the user parameter (either Device Related User Parameter or a List of Module Related User Parameter or both) to be verified by the application of the DP-Slave.

9.2.5.3.2 Check User Prm Result

This service is used in a DP-Slave as a reply to the previous issued Check User Prm service. The Default AP of a DP-Slave uses this service to provide the result of check User Prm Data.

Table 169 – Check User Prm Result

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Prm OK	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 used as local identifier of the desired AR.

Prm OK

The parameter Prm OK equal TRUE specifies that the User Prm Data conveyed by the previous Check User Prm service are acceptable by the application of the DP-Slave. If this parameterisation fails, i. e. this parameter has the value FALSE, the parameters Prm Req and Prm Fault in the Diag Data will be set.

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the cause of the failure.

Allowed values:

Table 170 – Status Values

Value	Meaning
Reset	Reset still in progress
Not Pending	No Check of User Prm Data pending
New Prm	Check of User Prm Data to be done

9.2.5.3.3 Check Cfg

The DP Application Layer of the DP-Slave notifies the application when Cfg Data are received from the DP-Master because this service requests a validation of the Cfg Data by the application. A further Check Cfg indication is only issued after the Check Cfg service is completed with a Check Cfg Result service. The DP-Slave compares its own configuration data with the Cfg Data received from the DP-Master.

The check shall cover the following details:

- module structure
- range of Input and/or Output Data Element
- format
- consistency.

The consistency also affects the DP-Slave and the DP-Master. The verification of the consistency in the Cfg Data list element shall cause only in the following cases a configuration fault:

- The DP-Slave needs consistency for an Input and/or Output Data Element and the DP-Master indicates no consistency.
- The DP-Slave can not provide consistency for an Input and/or Output Data Element and the DP-Master requires consistency for this Input and/or Output Data Element.

DP-Slaves may accept different sets of Cfg Data. In this case the Check Cfg shall be successful if one set of configuration data matches. If the service parameter Check Cfg Mode is set to TRUE the check of the Cfg Data may be done in a different User specific way. For example a temporary not available module may be accepted even if the Cfg Data contains the respective Configuration Identifier.

Table 171 – Check Cfg

Parameter name	.ind
Argument	M
AREP	M
Check Cfg Mode	C
Cfg Data	M

Argument

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

AREP

This parameter is used as local identifier of the AR.

Check Cfg Mode

This parameter shall only be present if the attribute DPV1 Enabled of the related MS0 AR is set to TRUE.

Cfg Data

This parameter contains the Cfg Data to be verified by the application of the DP-Slave.

9.2.5.3.4 Check Cfg Result

This service is used in a DP-Slave as a reply to the previous issued Check Cfg service. The Default AP of the DP-Slave uses this service to provide the result of check of the Cfg Data.

Table 172 – Check Cfg Result

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Cfg OK	M	
Input Data Len	M	
Output Data Len	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 used as local identifier of the desired AR.

Cfg OK

The parameter Cfg OK equal TRUE specifies that the Cfg Data conveyed by the previous Check Cfg service are acceptable by the application of the DP-Slave. The application of the DP-Slave may update the Real Cfg Data of the MS0 AR with the Set Cfg service.

If the check of the Cfg Data fails, i. e. this parameter has the value FALSE, the parameters Prm Req and Cfg Fault in the Diag Data will be set.

Input Data Len

This parameter contains the actual length of the Input Data.

Output Data Len

This parameter contains the actual length of the Output Data.

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the cause of the failure.

Allowed values:

Table 173 – Status Values

Value	Meaning
Reset	Reset still in progress
Not Pending	No Check of Cfg Data pending
New Cfg	Check of Cfg Data to be done

9.2.5.3.5 Set Cfg

This service is used by the application of the DP-Slave to pass its own configuration data to the DP-Application Layer for the MS0 AR of the DP-Slave.

Table 174 – Set Cfg

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Cfg 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 used as local identifier of the desired AR.

Cfg Data

This parameter contains the Cfg Data.

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

9.2.5.3.6 Get Cfg

This service allows the DP-Master (Class 2) to read the configuration data of the DP-Slave via the MS0 AR.

Table 175 – Get Cfg

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Result (+)		S
AREP		M
Cfg Data		M
Result (-)		S
AREP		M
Status		M

Argument

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

AREP

This parameter is used as local identifier of the desired AR.

Result (+)

This selection type parameter indicates that the service request succeeded.

Cfg Data

This parameter contains the Cfg Data.

Result (-)

This selection type parameter indicates that the service request failed.

Status

The parameter Status indicates the type of failure of the service.

Allowed values: DS, NA, RS, UE, NR, RE

9.2.5.3.7 Set Slave Add

This service permits a DP-Master (Class 2) to change the address of a DP-Slave and to pass remanent Slave Data via the MS0 AR. The Set Slave Add service is only successful if the Ident Number transmitted corresponds with the Ident Number of the DP-Slave and the attribute No Add Change (see 9.3.5.1.1) of the DP-Slave is FALSE.

Table 176 – Set Slave Add

Parameter name	.req	.ind	.cnf
Argument	M	M(=)	
AREP	M	M(=)	
New Slave Add	M	M(=)	
Ident Number	M	M(=)	
No Add Chg	M	M(=)	
Rem Slave Data	U	C(=)	
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 used as local identifier of the desired AR.

New Slave Add

This parameter contains the value of the attribute New Slave Add of the Remanent Parameter object of the DP-Slave which is to be set.

Ident Number

This parameter contains the Ident Number to be checked with the Ident Number of the DP-Slave.

No Add Chg

This parameter contains the value of the attribute No Add Change of the Remanent Parameter object. The attribute No Add Change will assume the value of this service parameter in case of success, otherwise No Add Change remains unchanged.

Rem Para

This parameter contains the value of the attribute Rem Para of the Remanent Parameter object.

Result (+)

This selection type parameter indicates that the service request succeeded. This confirmation does not reflect whether the service was successful, i. e. the new values were accepted by the DP-Slave. Therefore the DP-Master shall check the correct execution of the service by using the Read Slave Diag service with the new DP-Slave address.

Result (-)

This selection type parameter indicates that the service request failed.

Status

The parameter Status indicates the type of failure of the service.

Allowed values: DS, NA, RS, RR, UE, RE

9.2.5.3.8 Initiate

This service shall be used by a DP-Master (Class 2) to establish a connection to a desired DP-Slave.

Table 177 – Initiate

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
Send Timeout	M			
Features Supported	M	M(=)		
Profile Features Supported	M	M(=)		
Profile Ident Number	M	M(=)		
Add Addr Param	M	M(=)		
S-Type	M	M(=)		
S-Len	M	M(=)		
D-Type	M	M(=)		
D-Len	M	M(=)		
S-Addr	M	M(=)		
S-API	M	M(=)		
S-SCL	M	M(=)		
S-Network Address	C	C(=)		
S-MAC Address	C	C(=)		
D-Addr	M	M(=)		
D-API	M	M(=)		
D-SCL	M	M(=)		
D-Network Address	C	C(=)		
D-MAC Address	C	C(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Max Len Data Unit			M	M(=)
Features Supported			M	M(=)
Profile Features Supported			M	M(=)
Profile Ident Number			M	M(=)
Add Addr Param			M	M(=)
S-Type			M	M(=)
S-Len			M	M(=)
D-Type			M	M(=)
D-Len			M	M(=)
S-Addr			M	M(=)
S-API			M	M(=)
S-SCL			M	M(=)
S-Network Address			C	C(=)
S-MAC Address			C	C(=)

(continued on next page)

D-Addr			M	M(=)
D-API			M	M(=)
D-SCL			M	M(=)
D-Network Address			C	C(=)
D-MAC Address			C	C(=)
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.

Max Len Data Unit

This parameter contains the value of the attribute Max Len Data Unit of the ARL DP-Slave object of a DP-Slave.

Send Timeout

This parameter defines the control time for monitoring the MS2 AR requested by the DP-Master (Class 2). DP-Master and DP-Slave inform each other about the supported Send Timeout. The DP-Slave adjust its Send Timeout. The DP-Master (Class 2) decides whether the reported Send Timeout is sufficient or not.

Features Supported

DP-Master and DP-Slave inform each other about the supported service functionality. The DP-Slave has the possibility to adjust its functionality to the Master's requirements. The User application of the DP-Master has to decide if the reported functionality is sufficient or not.

Profile Features Supported

DP-Master and DP-Slave inform each other about the supported service functionality regarding the used profile definition, the profile is identified by the Profile Ident Number. The meaning of the defined bits is profile or vendor specific.

Profile Ident Number

This parameter identifies a profile definition uniquely. All devices using the same profile definition have to use the same Profile Ident Number. The Profile Ident Number will be taken from the pool of Ident Numbers for vendor specific or authorised profiles. The value 0 indicates that no profile is supported.

If the requested profile is supported by the DP-Slave, the Profile Ident Number is mirrored in the response. If the requested profile is not supported by the DP-Slave, the DP-Slave responds negatively or with a Profile Ident Number supported by the DP-Slave.

Add Addr Param

Add Addr Param consists of two parts, additional address information of the source and additional address information of the destination. The additional address information contains the parameter API plus an optional Network/MAC address.

D-Addr and S-Addr (API, SCL, Network Address, MAC Address) are related to the direction of the telegrams, which means the source becomes destination and vice versa in the response.

Add Addr Param contains the following subparameters:

S-Type

This subparameter indicates the presence (S-Type = 1) of the optional Network/MAC address in the Add Addr Param of the source.

S-Len

This subparameter indicates the length of the S-Addr subparameter.

D-Type

This subparameter indicates the presence (D-Type = 1) of the optional Network/MAC address in the Add Addr Param of the destination.

D-Len

This subparameter indicates the length of the D-Addr subparameter.

S-Addr

This subparameter contains the additional address information of the source:

S-API

This subparameter identifies the AP of the source.

S-SCL

This subparameter identifies the access level of the source.

S-Network Address

In case of S-Type = 1 this subparameter identifies the network address of the source according to ISO/OSI-Network addresses.

S-MAC Address

In case of S-Type = 1 this subparameter identifies the MAC Address of the source.

D-Addr

This subparameter contains the additional address information of the destination:

D-API

This subparameter identifies the AP of the destination.

D-SCL

This subparameter identifies the access level of the destination.

D-Network Address

In case of D-Type = 1 this subparameter identifies the network address of the destination according to ISO/OSI-Network addresses.

D-MAC Address

In case of D-Type = 1 this subparameter identifies the MAC Address of the destination.

Result (+)

This selection type parameter indicates that the service request succeeded.

Result (-)

This selection type parameter indicates that the service request failed.

Error Decode:

This parameter selects one of the following Error schemes: DPV1, FMS, Profile_Specific.

Error Code 1:

In case of Error Decode equal DPV1 the Error Code 1 assumes one of the following values: read error, write error, module failure, version conflict, feature not supported, user specific, invalid index, write length error, invalid slot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, write constrain conflict, resource busy, resource unavailable.

Error Code 2:

The parameter Error Code 2 is user specific.

9.2.5.3.9 Abort

The DP-Master or DP-Slave use this service to abort an AR. With the Abort.req the User resets the FSPM. In faulty conditions the FSPM informs the User with an Abort.ind and resets itself.

Table 178 – Abort

Parameter name	.req	.ind
Argument	M	M
AREP	M	M(=)
Locally Generated		C
Subnet	C	C(=)
Instance	C	C(=)
Reason Code	C	C(=)
Additional Detail		C

Argument

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

AREP

This parameter is used as local identifier of the desired AR.

Locally Generated

This parameter only exists in the indication in case of an MS2 AR. It indicates that the Abort has been produced by the remote partner (Locally Generated = FALSE) or in the local station (Locally Generated = TRUE).

Subnet

This parameter is used in case of an MS2 AR to encode information about the location of the source of the Abort.

Instance

This parameter is used in case of an MS2 AR to encode information about the protocol instance that detected a problem causing the Abort.

Allowed values:

Table 179 – Instance

Value	Meaning
DL	Data Link Layer
MS2	MS2 AR
User	User

Reason Code

The Reason Code indicates the reason for the Abort in case of an MS2 AR.

Allowed Values: UE, RR, RS, NR, DH, LR, RDL, RDH, DS, NA, ABT_SE, ABT_FE, ABT_TO, ABT_RE, ABT_IV, ABT_STO, ABT_IA, ABT_OC.

Additional Detail

This parameter contains detail information in case of an abort which is caused by an inappropriate Initiate (e.g. Send Timeout to small) in case of an MS2 AR. This parameter is only present if the parameter Reason Cde has the value ABT_STO and the parameter Instance equals MS2. In this case the parameter additional detail contains the value of the received Send Timeout of the DP-Slave.

9.2.5.3.10 DLL Init DP Slave

This service is used to initialise the data link layer of a DP-Slave.

Table 180 – DLL Init DP Slave

Parameter name	.req	.cnf
Argument	M	
Bus Para	M	
TS	M	
minTS DR	M	
Baudrate	M	
Slot Timer	M	
Result		M

Argument

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

Bus Para

This parameter is composed of the following subparameters:

TS

This parameter contains the DL-address of this DP-Slave.

minTSDR

This parameter contains the smallest station delay time of this DP-Slave.

Baudrate

This parameter contains the baudrate of this DP-Slave.

Slot Time

This parameter contains the slot time of this DP-Slave.

Result

This parameter indicates that the service request succeeded.

9.2.5.3.11 MS0 Init DP Slave

This service is used to initialise the MS0 AR of a DP-Slave.

Table 181 – MS0 Init DP Slave

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Result		M
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.

Result

This parameter indicates that the service request succeeded.

9.2.5.3.12 MS1 Init DP Slave

This service is used to initialise the MS1 AR of a DP-Slave.

Table 182 – MS1 Init DP Slave

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Result		M
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.

Result

This parameter indicates that the service request succeeded.

9.2.5.3.13 MS2 Init DP Slave

This service is used to initialise the MS2 AR of a DP-Slave.

Table 183 – MS2 Init DP Slave

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Result		M
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.

Result

This parameter indicates that the service request succeeded.

9.2.5.3.14 DP Slave Started

With this service the supported alarm features on this MS1 AR are indicated. The indication occurs after the DP-Slave has entered the data exchange mode and the transfer of alarm and status messages is possible from now on.

Table 184 – DP Slave Started

Parameter name	.ind
Argument	M
AREP	M
Actual Enabled Alarms	M
Alarm Sequence	M
Alarm Limit	C

Argument

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

AREP

This parameter is the local identifier for the desired AR.

Actual Enabled Alarms

This parameter indicates which alarm types are supported by the DP-Master (Class 1).

Allowed values: Update Alarm enabled / disabled, Manufacturer Specific Alarm enabled / disabled, Diagnosis Alarm enabled / disabled, Process Alarm enabled / disabled, Pull Plug Alarm enabled / disabled

Alarm Sequence

This parameter indicates whether only one alarm of a specific Alarm Type can be active at one time (Alarm Sequence=FALSE) or several alarms (2 to 32) of the same or different type can be active at one time (Alarm Sequence=TRUE).

Allowed values: TRUE, FALSE

Alarm Limit

This parameter indicates the maximum number of parallel alarms.

Allowed values:

Table 185 – Alarm Limit

Value	Meaning
0	
1	
2	
3	
4	
5	
6	
7	

9.2.5.3.15 DP Slave Stopped

In case of a an MS0 AR, this Application Layer service indicates to the User that the DP-Slave has left the data exchange mode and that the MS1 AR is closed. As a consequence the transfer of alarms as well as the processing of Write and Read services is no longer possible.

In case of a an MS1 AR, this service indicates to the User that the acyclic communication relationship to the DP-Master (Class 1) has been stopped.

Table 186 – DP Slave Stopped

Parameter name	.ind
Argument	M
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.

9.2.5.3.16 Reset DP Slave

By this service the User resets the Application Layer.

Table 187 – Reset DP Slave

Parameter name	.req	.cnf
Argument		

Argument

The argument shall convey the service specific parameters of the service request. Nevertheless, in this service there are no specific parameters.

9.2.5.3.17 DP Slave Fault

By this service the Application Layer indicates to the User that due to fatal errors all ARs have been reset.

Table 188 – DP Slave Fault

Parameter name	.ind
Argument	

Argument

The argument shall convey the service specific parameters of the service request. Nevertheless, in this service there are no specific parameters.

9.2.5.3.18 Application Ready DP Slave

By this service the User indicates to the Application Layer that the application is ready.

Table 189 – Application Ready DP Slave

Parameter name	.req
Argument	M
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.

9.2.5.3.19 Init DP Master C11

By this service the User initialises the Application Layer of a DP-Master (Class 1).

Table 190 – Init DP Master C11

Parameter name	.req	.cnf
Argument	M	
Bus Para	M	
TS	M	
Baud rate	M	
TSL	M	
minTSDR	M	
maxTSDR	M	
TQUI	M	
TSET	M	
TTR	M	
G	M	
HSA	M	
max retry limit	M	
Result		M

Argument

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

Bus Para

This parameter is composed of the following subparameters:

TS

This parameter contains the DL-address of this DP-Master (Class 1).

Baud rate

This parameter contains the data signaling rate of this DP-Master (Class 1).

TSL

This parameter contains the slot time for this DP-Master (Class 1).

minTSDR

This parameter contains the smallest station delay time for this DP-Master (Class 1).

maxTSDR

This parameter contains the largest station delay time for this DP-Master (Class 1).

TQUI

This parameter contains the transmitter fall time for this DP-Master (Class 1).

TSET

This parameter contains the setup time for this DP-Master (Class 1).

TTR

This parameter contains the target rotation time for this DP-Master (Class 1).

G

This parameter contains the GAP update factor for this DP-Master (Class 1).

HSA

This parameter contains the highest station (DL-) address for this DP-Master (Class 1).

max retry limit

This parameter contains the maximum number of retries for this DP-Master (Class 1).

Result

This parameter indicates that the service request succeeded.

9.2.5.3.20 DP Master CI1 Started

In case of a MS0 AR, this Application Layer service indicates to the User that the assigned DP-Slave is in the data exchange mode.

In case of a an MS1 AR, this Application Layer service indicates that the handling of alarm requests for this DP-Slave are possible.

Table 191 – DP Master CI1 Started

Parameter name	.ind
Argument	M
AREP	M
Alarm Limit	C

Argument

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

AREP

This parameter is the local identifier for the desired AR (Address of DP-Slave).

Alarm Limit

In case of a an MS1 AR, this parameter indicates the maximum number of parallel alarms.

Allowed values:

Table 192 – Alarm Limit

Value	Meaning
0	
1	
2	
3	
4	
5	
6	
7	

9.2.5.3.21 DP Master CI1 Stopped

This Application Layer service indicates that the related AR has been stopped. In case of a MS0 AR, the MS1 AR has been stopped, too.

Table 193 – DP Master CI1 Stopped

Parameter name	ind
Argument	M
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.

9.2.5.3.22 Reset DP Master CI1

By this service the User resets the Application Layer.

Table 194 – Reset DP Master CI1

Parameter name	.req	.cnf
Argument	M	
Result		M

Argument

The argument shall convey the service specific parameters of the service request. Nevertheless, in this service there are no specific parameters.

Result

This parameter indicates that the service request succeeded.

9.2.5.3.23 DP Master CI1 Fault

By this service the Application Layer indicates to the User that a fatal error has occurred and all ARs have been stopped. Further communication is only possible after a Reset.req of the User.

Table 195 – DP Master CI1 Fault

Parameter name	.ind
Argument	

Argument

The argument shall convey the service specific parameters of the service request. Nevertheless, in this service there are no specific parameters.

9.2.5.3.24 DP Master CI1 Reject

By this service the Application Layer rejects a previous Read/ Write.req to the MS1 AR. If the MS1 AR was forced to stop.

Table 196 – DP Master CI1 Reject

Parameter name	.ind
Argument	M
AREP	M
Reason	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 (Address of DP-Slave).

Reason

The parameter Status indicates success or failure of the service.

Allowed values: REJ ABORT, REJ LE, REJ PS, REJ SE, REJ IV

9.2.5.3.25 Set Mode DP Master CI1

By this service the User can set the operation mode of the MS0 AR. Additionally the User gets information about the actual state of the bus system in case of errors.

Table 197 – Set Mode DP Master Cl1

Parameter name	.req	.cnf
Argument	M	
AREP	M	
USIF State	M	
Result(+)		M
AREP		M
Bus Accessible		M
Result(-)		
AREP		M
Bus Accessible		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.

USIF State

The Application Layer provides four operation modes for the User:

- **OFFLINE**
Communication to all DP participants (DP-Master and DP-Slave) is stopped. Local DL control will be taken out of the token ring. The Application Layer waits for a signal to start.
- **STOP**
Changing the operation mode from OFFLINE to STOP the bus parameter set is loaded in the DL by the Application Layer. The DL control is set to active. The responder services for communication with the DP-Master (Class 2) are processed. DP-Slaves are not polled.
- **CLEAR**
In this operation mode the DP-Master (Class 1) attempts to configure the DP-Slaves that are assigned to it, and starts the data exchange mode. The input data of the DP-Slaves are read and passed to the User. The User's output data are ignored. In output direction zeros or no data are transmitted to the DP-Slaves.
- **OPERATE**
The DP-Master (Class 1) is in the data exchange mode with the assigned DP-Slaves. The inputs from the DP-Slaves are transferred to the User and the outputs from the User are passed to the DP-Slaves. If the Application Layer leaves the mode OPERATE, all DP-Slaves' outputs change in the safe state by means of the service Global Control (Control Command = Clear Data, Group Select = 0).

Result(+)

This selection type parameter indicates that the service request succeeded.

Bus Accessible

This parameter indicates whether the DP-Master is able to access the bus (TRUE) or not (FALSE).

Allowed values: TRUE, FALSE

Result(-)

This selection type parameter indicates that the service request failed.

9.2.5.3.26 DP Master CI1 Mode Changed

By this service the Application Layer indicates:

- the change of its operation mode from OPERATE to CLEAR if an error occurred and the Error Action Flag is set
- the change of its operation mode from STOP to OFFLINE if an error occurred while setting bus parameters.

Table 198 – DP Master CI1 Mode Changed

Parameter name	.ind
Argument	M
AREP	M
USIF State	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.

USIF State

Allowed values: OFFLINE, CLEAR (see service Set Mode)

9.2.5.3.27 Load Bus Par DP Master CI1

By this service the User can load a new bus parameter set. This service is not allowed in the operation mode OFFLINE. During STOP changes to all bus parameters are possible. In the operation modes CLEAR and OPERATE of the Application Layer it is not allowed to load a bus parameter set with changes to critical parameters. These critical parameters are the DL parameters Baud Rate and DL Add.

The changed bus parameters are not stored. Thus, all parameters of this service are lost after changing the operation mode of the Application Layer from STOP to OFFLINE. Changes to be stored can only be done within the bus parameter set (Bus Para) of the Init service.

Table 199 – Load Bus Par DP Master CI1

Parameter name	.req	.cnf
Argument	M	
Bus Para	M	
Result(+)		S
Status		M
Result(-)		S
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.

Bus Para

This parameter contains the bus parameter set which shall be loaded. The structure is similar to the current bus parameter set (see clause “Coding of the Bus Parameter Set”).

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Status

The parameter Status indicates the reason for success or failure of the service.

Allowed values: NO, IV

9.2.5.3.28 Mark DP Master CI1

By this service the User can operate synchronously with the MS0 AR. After the User has sent this service to the Application Layer, the User will receive an acknowledgement from the Application Layer after all activated DP-Slaves were polled at least once.

Table 200 – Mark DP Master CI1

Parameter name	.req	.cnf
Argument	M	
AREP	M	
Result(+)		S
AREP		M
Dia		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.

Result(+)

This selection type parameter indicates that the service request succeeded.

Dia

This parameter indicates (if the value is equal TRUE) whether at least one activated DP-Slave was not in the User data exchange mode.

Allowed values: TRUE, FALSE

Result(-)

This selection type parameter indicates that the service request failed.

Status

The parameter Status indicates success or failure of the service. The value NO will be signalled if the Application Layer is in the operation mode OFFLINE or STOP, or a previous Mark.req is still active.

Allowed values: NO

9.2.5.3.29 Abort DP Master Cl1

This service allows to release a MS0, MS1 or MM1 AR of a DP-Master (Class 1). The indication may be issued by the MS1 AR if internal errors occurred in this case the MS0 AR will also be released.

Table 201 – Abort DP Master Cl1

Parameter name	.req	.ind
Argument	M	M
AREP	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.

9.2.5.3.30 Read Value DP Master Cl1

This service allows to read any of the DL variables.

Table 202 – Read Value DP Master Cl1

Parameter name	.req	.cnf
Argument	M	
Variable	M	
Result		M
Value		M
Status		M

Argument

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

Variable

The parameter Variable selects the DL Variable whose current value has to be read. See Part 3 for a list of all possible DL variables.

Result

This parameter indicates that the service request succeeded.

Value

This parameter contains the actual value of the read DL Variable.

Status

The parameter Status indicates success or failure of the service.

Allowed values: OK, NO, IV

9.2.5.3.31 Delete SC DP Master C11

By this service the statistic counters can be cleared.

Table 203 – Delete SC DP Master C11

Parameter name	.req	.cnt
Argument	M	
Address	M	
Result		M

Argument

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

Address

This parameter specifies the DL address of the statistic counters which have to be cleared.

Allowed values: 0 to 125

Result

This parameter indicates that the service request succeeded.

9.2.5.3.32 DP Master C11 Event

By this service the events of the Service DLM-Event are indicated to the User as an Event.

Table 204 – DP Master C11 Event

Parameter name	.ind
Argument	M
Event	M
Add Info	M

Argument

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

Event

This parameter can take the values which are allowed for the DLM-Event service.

Add Info

The parameter indicates the added info of the DL.

9.2.5.3.33 Init DP Master Cl2

By this service the User initialises the DL-parameter of a DP-Master (Class 2).

Table 205 – Init DP Master Cl2

Parameter name	.req	.cnf
Argument	M	
Bus Para	M	
TS	M	
Baud rate	M	
TSL	M	
minTSDR	M	
maxTSDR	M	
TQUI	M	
TSET	M	
TTR	M	
G	M	
HSA	M	
max retry limit	M	
Result		M

Argument

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

Bus Para

This parameter is composed of the following subparameters:

TS

This parameter contains the DL-address of this DP-Master (Class 2).

Baud rate

This parameter contains the data signaling rate of this DP-Master (Class 2).

TSL

This parameter contains the slot time for this DP-Master (Class 2).

minTSDR

This parameter contains the smallest station delay time for this DP-Master (Class 2).

maxTSDR

This parameter contains the largest station delay time for this DP-Master (Class 2).

TQUI

This parameter contains the transmitter fall time for this DP-Master (Class 2).

TSET

This parameter contains the setup time for this DP-Master (Class 2).

TTR

This parameter contains the target rotation time for this DP-Master (Class 2).

G

This parameter contains the GAP update factor for this DP-Master (Class 2).

HSA

This parameter contains the highest station (DL-) address for this DP-Master (Class 2).

max retry limit

This parameter contains the maximum number of retries for this DP-Master (Class 2).

Result

This parameter indicates that the service request succeeded.

9.2.5.3.34 Reset DP Master CI2

By this service the User resets the Application Layer.

Table 206 – Reset DP Master CI2

Parameter name	.req	.cnf
Argument		

Argument

The argument shall convey the service specific parameters of the service request. Nevertheless, in this service there are no specific parameters.

9.2.5.3.35 DP Master CI2 Fault

By this service the Application Layer indicates a fatal DL error to the User.

Table 207 – DP Master CI2 Fault

Parameter name	.ind
Argument	

Argument

The argument shall convey the service specific parameters of the service request. Nevertheless, in this service there are no specific parameters.

9.2.5.3.36 DP Master CI2 Reject

By this service the Application Layer rejects a previous Read, Write, Data Transport service if the MS2 AR was forced to stop.

Table 208 – DP Master CI2 Reject

Parameter name	.ind
Argument	M
AREP	M
Reason	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 (Address of DP-Slave).

Reason

The parameter Status indicates success or failure of the service.

Allowed values: REJ ABORT, REJ LE, REJ PS, REJ SE, REJ IV

9.2.5.3.37 DP Master CI2 Closed

By this service the Application Layer informs the User that the MS2 AR has been closed.

Table 209 – DP Master CI2 Closed

Parameter name	.ind
Argument	M
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.

9.2.5.3.38 DP Master CI2 Event

By this service the events of the Service DLM-Event are indicated to the User as an Event.

Table 210 – DP Master CI2 Event

Parameter name	.ind
Argument	M
Event	M
Add Info	M

Argument

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

Event

This parameter can take the values which are allowed for the DLM-Event service.

Add Info

The parameter indicates the added info of the DL.

9.2.6 Management ASE

9.2.6.1 Overview

The Management ASE is used to convey parameters between a DP-Master (Class 2) and a DP-Master (Class 1). Additionally, the DP-Master (Class 2) can read diagnosis information of the DP-Master (Class 1) and its assigned DP-Slaves. The access of the DP-Master (Class 2) to the DP-Master (Class 1) is performed according to the client/server access model.

9.2.6.2 Management Class Specification

9.2.6.2.1 Master Diag Class Specification

A Master Diag object is described by the following template:

DP ASE:	Management ASE
CLASS:	Master Diag
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m) Key Attribute:	Implicit
2. (m) Attribute:	Master Status
2.1. (m) Attribute:	USIF State
2.2. (m) Attribute:	Ident Number
2.3. (m) Attribute:	Hardware Release DP
2.4. (m) Attribute:	Firmware Release DP
2.5. (m) Attribute:	Hardware Release User
2.6. (m) Attribute:	Firmware Release User
3. (m) Attribute:	System Diagnosis
4. (m) Attribute:	Data Transfer List
5. (o) Attribute:	List of DP-Slave Diag
5.1 (m) Attribute:	DP-Slave Address
5.2 (m) Attribute:	DP-Slave Diag
SERVICES:	
1. (m) OpsService:	Get Master Diag

9.2.6.2.1.1 Attributes

Implicit

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

Master Status

This parameter is composed of the following items.

USIF State

This attribute describes the state of a DP-Master (Class 1)

Allowed values:

Table 211 – USIF State

Value	Meaning
0	OFFLINE
1	STOP In this state the DP-Master (Class 1) does not execute any services at the MS0 and MS1 AR. The MM1 and MM2 AR can be established.
2	CLEAR The DP-Master is operating the MS0 AR. The Output Data are in the safe state. The services related with MS1 AR can be executed. The MM1 and MM2 AR can be established.
3	OPERATE The DP-Master is operating the MS0 AR. The Output Data are valid. The services related with MS1 AR can be executed. The MM1 and MM2 AR can be established.

Ident Number

This attribute reflects the Ident Number of the DP-Master (Class 1).

Attribute Type: Unsigned16

Hardware Release DP

This attribute specifies the hardware release of the DP implementation of the DP-Master (Class 1).

Attribute Type: Unsigned8

Firmware Release DP

This attribute specifies the firmware/software release of the DP implementation of the DP-Master (Class 1).

Attribute Type: Unsigned8

Hardware Release User

This attribute specifies the hardware release of the User implementation of the DP-Master (Class 1).

Attribute Type: Unsigned8

Firmware Release User

This attribute specifies the firmware/software release of the User implementation of the DP-Master (Class 1).

Attribute Type: Unsigned8

System Diagnosis

The System Diagnosis reflects the state of all DP-Slaves assigned to the DP-Master (Class 1). This state itself is a one-bit-information for each DP-Slave which is set to the value 1 if its Ext Diag Flag of the Diagnosis information is set or if it does not respond or if it responds incorrectly so that the MS0 AR has to be re-established. The state of a DP-Slave is reset to the value 0 after the MS0 AR is successfully re-established or if the DP-Slave is deactivated by the AP of the DP-Master (Class 1) or the USIF State of the DP-Master (Class 1) is STOP.

Data Transfer List

The Data Transfer List reflects the status of the data exchange mode of all DP-Slaves assigned to the DP-Master (Class 1).

This status itself is a one-bit-information for each DP-Slave which is set to the value 1 if the data exchange with the assigned DP-Slave was successfully or if between two data exchange requests is only one diagnosis request. All other cases reset the status bit-information of the DP-Slave to the value 0.

List of DP-Slave Diag

This attribute is composed of the following list elements.

DP-Slave Address

This attribute specifies the DL address of the DP-Slave

Allowed values: 0 to 125

DP-Slave Diag

This attribute reflects the diagnosis information of the DP-Slave inclusive the information added by the DP Application Layers for each DP-Slave.

9.2.6.2.1.2 Invocation of Master Diag object

For the invocation of the Master Diag object the following rules have to be considered:

- Only one Master Diag object shall be invoked in a DP-Master (Class 1)

9.2.6.2.2 Master Parameter Class Specification

A Master Parameter object is described by the following template:

DP ASE:	Management ASE
CLASS:	Master Parameter
CLASS ID:	not used
PARENT CLASS:	TOP

ATTRIBUTES:

1. (m) Key Attribute: Implicit
2. (m) Attribute: Bus Para Len
3. (m) Attribute: TS
4. (m) Attribute: Baud rate
5. (m) Attribute: T_{SL}
6. (m) Attribute: min T_{SDR}
7. (m) Attribute: max T_{SDR}
8. (m) Attribute: T_{QUI}
9. (m) Attribute: T_{SET}
10. (m) Attribute: T_{TR}
11. (m) Attribute: G
12. (m) Attribute: HSA
13. (m) Attribute: max retry limit
14. (m) Attribute: Error Action Flag
15. (m) Attribute: Min Slave Interval
16. (m) Attribute: Poll Timeout
17. (m) Attribute: Data Control Time
18. (m) Attribute: Alarm Max
19. (m) Attribute: Max User Global Control
20. (m) Attribute: Master User Data Len
21. (m) Attribute: Master Class2 Name
22. (m) Attribute: Master User Data
23. (o) Attribute: Slave Para
 - 23.1. (m) Attribute: Slave DL address
 - 23.2. (m) Attribute: Slave Para Len
 - 23.3. (m) Attribute: Extra Alarm SAP
 - 23.4. (m) Attribute: DPV1 Data Types
 - 23.5. (m) Attribute: DPV1 Supported
 - 23.6. (m) Attribute: Fail Safe
 - 23.7. (m) Attribute: Active
 - 23.8. (m) Attribute: Slave Type
 - 23.9. (m) Attribute: New Prm
 - 23.10. (m) Attribute: Max Diag Data Len
 - 23.11. (m) Attribute: Max Alarm Len
 - 23.12. (m) Attribute: Max Channel Data Length
 - 23.13. (m) Attribute: Diag Upd Delay
 - 23.14. (m) Attribute: Alarm Mode

23.15.	(m)	Attribute:	NA to Abort
23.16.	(m)	Attribute:	Ignore ACLr
23.17.	(m)	Attribute:	Prm Data Len
23.18.	(m)	Attribute:	Prm Data
23.19.	(m)	Attribute:	Cfg Data Len
23.20.	(m)	Attribute:	Cfg Data
23.21.	(m)	Attribute:	Add Tab Len
23.22.	(m)	Attribute:	Add Tab
23.23.	(m)	Attribute:	Slave User Data Len
23.24.	(o)	Attribute:	Slave User Data

SERVICES:

1. (o) OpsService: Start Seq
2. (o) OpsService: Download
3. (o) OpsService: Upload
4. (o) OpsService: End Seq
5. (o) OpsService: Act Para Brct
6. (o) OpsService: Act Param

9.2.6.2.2.1 Attributes**Implicit**

This attribute indicates that the Master Parameter object is implicitly addressed by the services.

Bus Para Len

This attribute contains the length of Bus Para inclusive the length parameter.

Attribute Type: Unsigned16

Allowed values: 66 to $2^{16}-1$

TS

This attribute contains the own DL address of the DP-Master.

Attribute Type: Unsigned8

Allowed values: 0 to 125

Baud rate

This attribute contains the code number of the chosen baud rate.

Attribute Type: Unsigned8

Allowed values:

Table 212 – Baud rate

Value	Meaning
0	9,6kbit/s
1	19,2kbit/s
2	93,75kbit/s
3	187,5kbit/s
4	500 kbit/s
6	1500 kbit/s
7	3000 kbit/s
8	6000 kbit/s
9	12000 kbit/s
10	31,25kbit/s
11	45,45kbit/s

T_{SL}, min T_{SDR}, max T_{SDR}, T_{QUI}, T_{SET}, T_{TR}, G, HSA, max retry limit

These attributes are described in the Type 3 Data Link Layer Service and Protocol Specification (see part 3 and part 4 of this International Standard).

Error Action Flag

This attribute reflects the reaction of the DP-Master (Class 1) in case of failed data transfers. This attribute will be evaluated in the operation mode OPERATE and CLEAR of the DP-Master (Class 1). If the data transfer to at least one activated DP-Slave was impossible for a certain period of time (Data Control Time) and if the Error Action Flag is set, the operation mode will change from OPERATE to CLEAR. If the Error Action Flag is cleared, the DP-Master (Class 1) will remain in the operation mode OPERATE despite erroneous data transfers.

If the operation mode of the DP-Master (Class 1) shall be changed from CLEAR to OPERATE with the Error Action Flag set to TRUE the data transfer to all activated DP-Slaves have to be performed successfully for a certain period of time (Data Control Time)

Allowed values:

Table 213 – USIF State

Value	Meaning
0	no change of the operation mode in case of an error
1	change of the operation mode in case of an error

Min Slave Interval

This attribute specifies the smallest allowed period of time between two consecutive DP-Slave poll cycles. This ensures that the sequence of cyclic service requests from the DP-Master (Class 1) can be handled by the DP-Slave. This period of time will be complied by the DP-Master (Class 1) for every cyclic Master-Slave service except for the Global Control service. For the Global Control service the User is responsible for compliance with Min Slave Interval.

Attribute Type: Unsigned16

Allowed values: 1 to $2^{16}-1$

Time Base: 100µs

Poll Timeout

In the case of Master-Master communication this attribute specifies the maximum period of time it may take the requester to fetch the response.

Attribute Type: Unsigned16

Allowed values: 1 to $2^{16}-1$

Time Base: 1ms

Data Control Time

This attribute specifies the period of time during a Global Control will be send and at least one Data Exchange should be executed with every activated DP-Slave.

Attribute Type: Unsigned16

Allowed values: 1 to $2^{16}-1$

Time Base: 10ms

Alarm Max

This variable contains the maximum number of alarms per DP-Slave, which can be handled by the DP-Master. It describes a Master capability. The Master shall be able to handle a minimum number of 7 alarms, because there are existing 7 different alarm classes.

Attribute Type: Unsigned8

Allowed values: 7 to 32

Max User Global Control

This attribute defines the maximum number of Global Control requests, which may be started by the User at the same time. This parameter describes the ability of the DP-Master (Class 1). A practicable value for Max User Global Control is 16. For each of the 8 Slave groups one Sync and one Freeze Command may be started at the same time.

Attribute Type: Unsigned8

Allowed values: 1 to 255

Master User Data Len

This attribute contains the length of Master User Data inclusive the length parameter.

Attribute Type: Unsigned16

Allowed values: 34 to $2^{16}-1$

Master Class2 Name

This attribute indicates the name of the DP-Master (Class 2) the Master parameter set where created with.

Attribute Type: Visible String(32)

Master User Data

This attribute contains manufacturer specific data which is necessary for the bus parameter set.

Attribute Type: Octet String

Slave Para

This combined attribute consists of the following items.

Slave DL address

This attribute reflects the DL address of the DP-Slave. This address is used as an identifier (Area Code) to address the corresponding slave parameter set, i.g. for up- or download purposes.

Attribute Type: Unsigned8

Allowed values: 0 to 125

Slave Para Len

This attribute contains the length of Slave Para inclusive the length parameter. A DP-Slave parameter set can be deleted by setting the Slave Para Len to zero.

Attribute Type: Unsigned16

Allowed values: 0 to $2^{16}-1$

Extra Alarm SAP

This attribute selects the Channel for the Alarm Ack service.

If the attribute is set to FALSE the DP-Master (Class 1) acknowledges alarms via SAP51.

If the attribute is set to TRUE the DP-Master (Class 1) acknowledges alarms via SAP50.

Allowed values: TRUE, FALSE

DPV1 Data Types

This attribute reflects whether the DP-Slave uses data types.

If the attribute is set to FALSE the DP-Slave uses no data types for the Data Description.

If the attribute is set to TRUE the DP-Slave uses data types for the Data Description.

Attribute Type: Boolean

DPV1 Supported

This attribute reflects whether the DP-Slave uses the extended functionality (see 9.1.3.2.1).

If the attribute is set to FALSE the DP-Slave uses no extended functionality.

If the attribute is set to TRUE the DP-Slave uses extended functionality.

Attribute Type: Boolean

Fail Safe

This attribute reflects whether the DP-Slave supports the Fail Safe functionality. If the attribute is set to FALSE the DP-Slave does not support the Fail Safe functionality. If the attribute is set to TRUE the DP-Slave supports the Fail Safe functionality.

Attribute Type: Boolean

Active

This attribute reflects whether the DP-Slave has to be activated. If the attribute Active has the value FALSE, the DP-Master (Class 1) stops the DP-Slave handling to this assigned DP-Slave. In this state it is allowed to change the DP-Slave parameter set, for example by a DP-Master (Class 2) with the Download service. If the Active-Flag has the value TRUE, the DP-Slave handling will be done by the DP-Master (Class 1) to this assigned DP-Slave. Changing of the DP-Slave parameter set in this state is limited (see also New Prm).

Attribute Type: Boolean

Slave Type

This attribute contains a manufacturer specific type denotation for the DP-Slave see Table 214.

Attribute Type: Unsigned8

Allowed values:

Table 214 – Slave Type

Value	Meaning
0	DP-Slave
1-15	reserved
16-255	manufacturer specific

New Prm

This attribute reflects whether parameterisation data has to be transferred to the DP-Slave in the data exchange mode. If this attribute is set to FALSE the Output Data will be transferred to this DP-Slave in the data exchange mode. If this attribute is set to TRUE, for one data cycle parameterisation data will be transferred instead of Output Data. The consistency of the parameterisation data has to be guaranteed by local measures. After transmitting this parameterisation data successfully the DP-Master (Class 1) sets this attribute to FALSE and the data exchange mode to the DP-Slave will be continued.

Attribute Type: Boolean

Max Diag Data Len

This attribute contains the maximum length of Diag Data for the DP-Slave.

Attribute Type: Unsigned8

Allowed values: 6 to 244

Max Alarm Len

This attribute specifies the maximum length of the Alarm within the DP-Slave Diag-PDU.

Attribute Type: Unsigned8

Allowed values: 4 to Min(Max Diag Data Len-6, 64)

Max Channel Data Length

This attribute defines the maximum length of a MS1-PDU (Length parameter and 4 Byte Header) for the corresponding DP-Slave.

Attribute Type: Unsigned8

Allowed values: 4 to 244

Diag Upd Delay

The attribute reflects the number of Diag service requests performed in the state DIAG2 of the DP-Master (Class 1) while Prm Req is set in Diag Data of the DP-Slave (for DP-Slaves with reduced performance).

Attribute Type: Unsigned8

Allowed values: 0 to 15 (extendible up to 255)

Alarm Mode

This attribute specifies the maximum number of possible active alarms.

Attribute Type: Unsigned8

Allowed values:

Table 215 – Alarm Mode

Value	Meaning
0	1 alarm of each type
1	2 alarms in total
2	4 alarms in total
3	8 alarms in total
4	12 alarms in total
5	16 alarms in total
6	24 alarms in total
7	32 alarms in total

NA To Abort

This attribute reflects the behaviour of the DP-Master (Class 1) if the assigned DP-Slave does not respond at the MS0 AR in the DIAG2 state and the data exchange mode.

If this attribute is set to FALSE the DP-Master continues in case of no response (Status = NA) of the assigned DP-Slave.

If this attribute is set to TRUE the DP-Master aborts the MS0 AR in case of no response (Status = NA) of assigned the DP-Slave (abort means sending an unlock command and to re-establish the MS0 AR).

Attribute Type: Boolean

Ignore ACIr

This attribute in combination with the Error Action Flag allows a DP-Slave specific reaction of the DP-Master (Class 1) in case of failed data transfers.

If the attribute is set to FALSE the DP-Master (Class 1) processes the auto-clear function as defined in the Error Action Flag.

If the attribute is set to TRUE the DP-Master (Class 1) ignores the auto-clear function as defined in the Error Action Flag.

Attribute Type: Boolean

Prm Data Len

This attribute contains the length of Prm Data inclusive the length parameter.

Attribute Type: Unsigned8

Allowed values: 9 to 246

Prm Data

This attribute reflects the parameterisation data send to the DP-Slave with the Set Prm service.

Cfg Data Len

This attribute contains the length of Cfg Data inclusive the length parameter.

Attribute Type: Unsigned8

Allowed values: 3 to 246

Cfg Data

This attribute reflects the Cfg Data send to the DP-Slave.

Add Tab Len

This attribute contains the length of Add Tab inclusive the length parameter.

Attribute Type: Unsigned8

Allowed values: 2 to $2^{16}-31$

Add Tab

This attribute contains the address assignment table of the DP-Slave. In case of a PLC this list contains the PLC addresses of the decentralized addresses.

Slave User Data Len

This attribute contains the length of the Slave User Data inclusive the length parameter.

Attribute Type: Unsigned8

Allowed values: 2 to $2^{16}-31$

Slave User Data

This attribute contains manufacturer specific data that characterise the DP-Slave for the DP-Master (Class 1).

Attribute Type: Octet String

9.2.6.2.2 Invocation of Master Parameter object

For the invocation of the Master Parameter object the following rules have to be considered:

- Only one Master Parameter object can be invoked in a DP-Master.
- Up to 125 Slave Para elements can be invoked in a DP-Master.

9.2.6.3 Management Service Specification

9.2.6.3.1 Get Master Diag

With this service it is possible to read Diagnosis Data of the DP-Master (Class 1) itself or of the DP-Slaves assigned to the DP-Master (Class 1). If the Diagnosis information of a DP-Slaves is requested the Diagnosis information stored in the DP-Master (Class 1) are used. The Diagnosis Data are structured in a system wide Diagnosis information collection and in a DP-Slave related Diagnosis information. With the MDiag Identifier parameter of the Get Master Diag service the different Diagnosis Data can be selected.

Table 216 – Get Master Diag

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
AREP	M	M(=)		
MDiag Identifier	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Diagnosis Data			M	M(=)
DP-Slave Diag			S	S(=)
System Diagnosis			S	S(=)
Data Transfer List			S	S(=)
Master Status			S	S(=)
Result (-)			S	S(=)
AREP			M	M(=)
Status			M	M(=)

Argument

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

AREP

This parameter is the local identifier of the desired AR.

MDiag Identifier

This parameter specifies the type of the requested Diagnosis Data or the DL address of a DP-Slave which Diagnosis information to be read.

Allowed values:

Table 217 – MDiag Identifier

Value	Meaning
0-125	Diag Data of the addressed DP-Slave
126	System Diagnosis
127	Master Status
128	Data Transfer List

Result (+):

This selection type parameter indicates that the service request succeeded.

Diagnosis Data

Different diagnosis information are transmitted depending on the MDiag Identifier.

DP-Slave Diag

This parameter contains the Diag Data of the addressed DP-Slave.

System Diagnosis

This parameter contain the System Diagnosis of the Master Diag object (see 9.2.6.2.1)

Master Status

This parameter contains the Master Status of the Master Diag object (see 9.2.6.2.1)

Data Transfer List

This parameter contains the Data Transfer List of the Master Diag (see 9.2.6.2.1)

Result (-):

This selection type parameter indicates that the service request failed.

Status

This parameter provide detailed information about the failure.

Allowed values: DS, NA, RS, RR, UE, RE, TO, FE, NE, IP, AD, EA, LE

9.2.6.3.2 Start Seq

With the Start Seq service the beginning of an Upload or Download sequence is signalled by the DP-Master (Class 2) in the DP-Master (Class 1). In this case the DP-Master (Class 1) can set an access protection for the addressed area until the End Seq service signals the end of the sequence. The addressed data can be transmitted in blocks with the services Upload and Download.

Table 218 – Start Seq

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M		
AREP	M	M		
Area Code	M	M(=)		
Timeout	M	M(=)		
Result (+)			S	S(=)
AREP			M	M
Max Len Data Unit			C	C
Result (-)			S	S(=)
AREP			M	M
Status			M	M

Argument

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

AREP

This parameter is the local identifier of the desired AP.

Area Code

The parameter Area Code identifies the area which shall be loaded or read. If the parameter Area Code is set to a value of 255, local access protection for a determined Area Code is not guaranteed for the services following. The local access protection ensures that the User cannot access the partially loaded data during the service sequence.

Type: Unsigned8

Allowed values:

Table 219 – Area Code

Value	Meaning
0 to 125	DP-Slave parameter set
126	reserved
127	Bus parameter set
128	reserved
129	statistic counters
130 to 135	reserved for diagnosis message filter
136 to 139	for the transfer of the Master parameter set
140 to 254	reserved
255	Start Seq: No local access protection

The User of the DP-Master (Class 1) has to provide an intermediate buffer which is only reserved for the transfer of the bus parameter set. The Download service transfers the bus parameter set into the intermediate buffer. By the services Act Param or Act Para Brct the bus

parameter set is transferred from the intermediate buffer to the active buffer. After the bus parameter set will be activated the contents of the intermediate buffer retain unchanged. The Upload service always reads from the active buffer.

Timeout

This parameter defines the control time between two successive Upload/Download services. If this time expires the access protection is deactivated. The previously transferred data are invalid.

Type: Unsigned16

Time Base: 1ms

Result (+):

This selection type parameter indicates that the service request succeeded.

Max Len Data Unit

The parameter Max Len Data Unit defines the maximum possible length of the parameter Data in the following Upload/Download frames.

Type: Unsigned8

Range: 1 to 240

Result (-):

This selection type parameter indicates that the service request failed.

Status

This parameter provide detailed information about the failure.

Allowed values: DS, NA, RS, RR, UE, TO, FE, RE, NE, AD, IP, NI, SE, SC, EA, LE

9.2.6.3.3 Download

This service is used to transfer the Master Parameter object from the DP-Master (Class 2) to a DP-Master (Class 1). The beginning of the Download sequence can be signalled with the service Start Seq. The actual data set is transmitted in blocks with the Download service in the addressed area. After the transfer is completed and possibly activated, the sequence shall be terminated with the End Seq service, if the Download sequence was started with the Start Seq service.

Table 220 – Download

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M		
AREP	M	M		
Area Code	M	M		
Add Offset	M	M		
Data	M	M		
Result (+)			S	S(=)
AREP			M	M
Result (-)			S	S(=)
AREP			M	M
Status			M	M(=)

Argument

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

AREP

This parameter is the local identifier of the desired AP.

Area Code

The parameter Area Code identifies the area which shall be loaded.

Type: Unsigned8

Allowed values: see 9.2.6.3.2

Add Offset

This parameter contains the offset to the begin of the area which is specified by the Area Code.

Type: Unsigned16

Data

This parameter contains the area data to be transferred.

Type: Octet String

Length: 1 to 240

Result (+):

This selection type parameter indicates that the service request succeeded.

Result (-):

This selection type parameter indicates that the service request failed.

Status

This parameter provide detailed information about the failure.

Allowed values: DS, NA, RS, RR, UE, TO, FE, RE, NE, AD, EA, LE, SC, NI, NC

9.2.6.3.4 Upload

This service is used to transfer the Master Parameter object from the DP-Master (Class 1) to a DP-Master (Class 2). The Upload sequence can be executed in the same way as the Download sequence. It is possible to transfer single data areas without the services Start Seq and End Seq. In this case the consistency of the transferred data is not guaranteed.

Table 221 – Upload

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M		
AREP	M	M		
Area Code	M	M		
Add Offset	M	M		
Data Len	M	M		
Result (+)			S	S(=)
AREP			M	M
Data			C	C(=)
Result (-)			S	S(=)
AREP			M	M
Status			M	M(=)

Argument

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

AREP

This parameter is the local identifier of the desired AR.

Area Code

The parameter Area Code identifies the area which shall be read

Type: Unsigned8

Allowed values: see 9.2.6.3.2

Add Offset

This parameter identifies the offset to the begin of the area which is specified by the Area Code.

Type: Unsigned16

Data Len

The parameter defines the length of the requested data.

Type: Unsigned8

Allowed values: 1 to 240

Result (+):

This selection type parameter indicates that the service request succeeded.

Data

This parameter contains the area data to be transferred.

Type: Octet String

Length: 1 to 240

Result (-):

This selection type parameter indicates that the service request failed.

Status

This parameter provide detailed information about the failure.

Allowed values: DS, NA, RS, RR, UE, TO, FE, RE, NE, EA, LE, NI, SC, AD

9.2.6.3.5 End Seq

This service is used to mark the end of a sequence of consistent transfers of data blocks of the Master Parameter object.

Table 222 – End Seq

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M		
AREP	M	M		
Result (+)			S	S(=)
AREP			M	M
Result (-)			S	S(=)
AREP			M	M
Status			M	M(=)

Argument

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

AREP

This parameter is the local identifier of the desired AR.

Result (+):

This selection type parameter indicates that the service request succeeded.

Result (-):

This selection type parameter indicates that the service request failed.

Status

This parameter provide detailed information about the failure.

Allowed values: DS, NA, RS, RR, UE, TO, FE, RE, NI, SE, NE, AD, EA, LE, NC

9.2.6.3.6 Act Para Brct

This service is used to activate or deactivate a parameter set by the DP-Master (Class 2). This service can be performed simultaneously to one (single) or many (multicast) DP-Master (Class 1) depending on the parameter D_addr. In the operation modes CLEAR and OPERATE of the DP-Master (Class 1), the activation of the bus parameter set with changed DL parameters Baud Rate or DL Add will be rejected. This service is unconfirmed because a possible change of the baud rate shall take effect at all addressed DP-Masters (Class 1) nearly at the same time.

Table 223 – Act Para Brct

Parameter name	.req	.ind	.cnf
Argument	M	M	
AREP	M	M	
Area Code	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 of the desired AR.

Area Code

The parameter Area Code identifies the area which shall be activated.

Type: Unsigned8

Range:

0 to 126	not allowed
127	bus parameter set
128 to 129	reserved
130 to 135	reserved for diagnosis message filter
136 to 139	for the activation of the Master parameter object
140 to 254	reserved
255	not allowed

In the presence of several DP-Masters (Class 2) it shall be guaranteed that the old and the new bus parameter sets are compatible.

Result (+):

This selection type parameter indicates that the service request succeeded.

Result (-):

This selection type parameter indicates that the service request failed.

Status

This parameter provide detailed information about the failure.

Allowed values: DS

9.2.6.3.7 Act Param

This service is used to activate or deactivate the Master Parameter object.

By this service a DP-Master (Class 2) is able

- to activate or to deactivate a parameterised DP-Slave
- to change the operation mode of the DP Application Layer
- to accept and activate the bus parameter set.

If the loaded bus parameter set contains a changed baud rate or station address for the DP-Master (Class 1) the new bus parameter set shall be activated with the Act Para Brct service.

Table 224 – Act Param

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M		
AREP	M	M		
Area Code	M	M(=)		
Activate	M	M(=)		
Result (+)			S	S(=)
AREP			M	M
Result (-)			S	S(=)
AREP			M	M
Status			M	M(=)

Argument

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

AREP

This parameter is the local identifier of the desired AR.

Area Code

The parameter Area Code identifies the area which shall be activated/deactivated with the exception of the operation mode.

Type: Unsigned8

Allowed values:

Table 225 – Area Code

Value	Meaning
0 to 125	DP-Slave parameter set The (Flag/Bit) Active in the DP-Slave parameter set of the DP-Master (Class 1) is influenced accordingly. The DP-Slave concerned takes part in the cyclic data exchange mode or is removed from this mode and no longer addressed.
126	reserved
127	bus parameter set
128	operation mode (description see clause “Set Operation Mode”)
129	reserved
130 to 135	reserved for diagnosis messages filter
136 to 139	activation of the Master parameter set
140 to 255	reserved

Activate

The meaning and the allowed values of this parameter are depending on the Area Code.

Type: Unsigned8

Table 226 – Activate

Area Code	Allowed values for the parameter Activate	
0 to 125: (DP-Slave parameter set)	0	(= Deactivate)
	128	(= Activate)
127: (bus parameter set)	255	(= Activate)
128: (operation mode)	64	(= STOP)
	128	(= CLEAR)
	192	(= OPERATE)

Result (+):

This selection type parameter indicates that the service request succeeded.

Result (-):

This selection type parameter indicates that the service request failed.

Status

This parameter provide detailed information about the failure.

Allowed values: DS, NA, RS, RR, UE, TO, FE, RE, NE, AD, IP, SC, NI, DI, EA, LE

9.2.7 AR ASE

9.2.7.1 Overview

This ASE specifies the structure of the ARs including the necessary DL parameters for the specific CRs. The AR ASE in each device manages an end point of the AR and all related CRs, and maintains their local context.

An AR is characterised by all of the following basic roles of the AREP used in Type 3 DP:

AR cardinality

- one-to-one
 - In a one-to-one AR an AP communicates with exactly one remote AP.
- one-to-many
 - In a one-to-many AR the AP of a DP-Master (Class 2) communicates with the APs of a group of DP-Masters (Class 1). This comprises the simultaneous communication to the defined group of DP-Masters (Class 1). Exactly one AP is addressed per device.
- many-to-many
 - In a many-to-many AR the APs of a DP-Master (Class 1) and optional the APs of one or several DP-Master (Class 2) communicates with the APs of a group of DP-Slaves. This comprises the sequential as well as the simultaneous communication to the defined group of DP-Slaves. Exactly one AP is addressed per device.

Device type relations:

Master-Slave (MS) communication

- between DP-Master (Class 1) and DP-Slaves
- between DP-Master (Class 2) and DP-Slaves

Master-Master (MM) communication

- between DP-Master (Class 2) and DP-Master (Class 1)

9.2.7.1.1 Communication Relationship

An AR is mapped to one or more communication relationships (CRs). A CR carries out the exchange of information based on a set of rules.

9.2.7.1.1.1 Communication Relationship Endpoint

CRs are defined as a set of co-operating Communication Relationship End Points (CREPs). The local context of an CR endpoint is used to control the conveyance of APDUs on the CR.

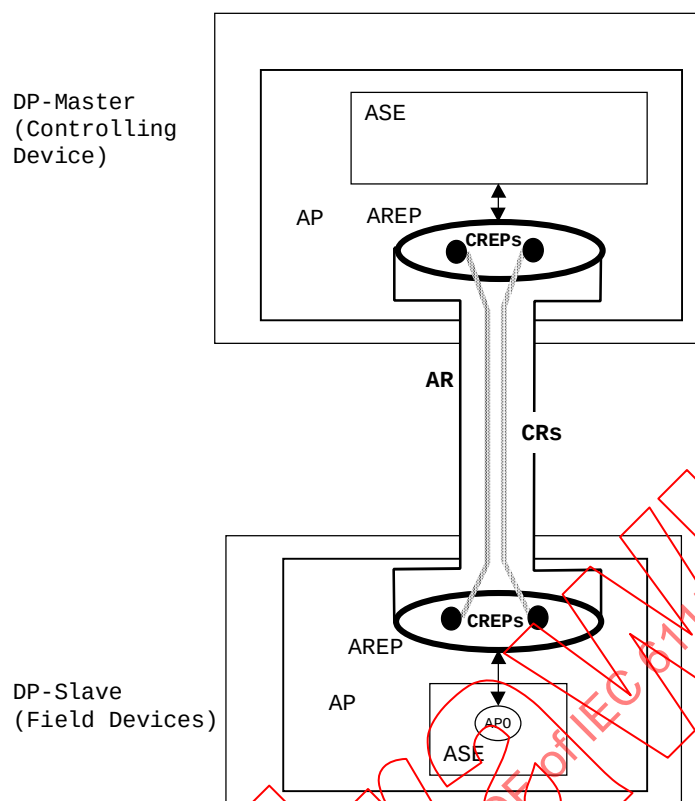


Figure 51 – Assignment of Communication Relationship to Application Relationship

9.2.7.1.1.2 Buffer and Queues

CREPs may be modelled as a queue or as a buffer. Application data transferred over a queued CREP are delivered in the order received for conveyance. The transfer of application data over a buffered CREP is different. In this case, application data is placed in an application layer buffer for transfer.

When the CREP receives another conveyance request, it replaces the previous contents of the buffer whether or not transmitted. Once application data are written into a buffer for transfer, it is preserved in the buffer until the next application data to be transmitted replaces it. While in the buffer, application data may be read more than once without deleting it from the buffer or changing its contents.

At the receiving end, the operation is similar. The receiving application layer places the received application data into a buffer for access by the different ASEs. When subsequent application data are received, it overwrites the previous application data in the buffer whether or not it was read by the corresponding ASE. Reading the application data from the buffer is not destructive it does not destroy or change the contents of the buffer, allowing the contents to be read from the buffer one or more times.

9.2.7.1.1.3 Communication Relationship Endpoint Roles

An communication relationship is characterised by all of the following basic roles of the CREP used in Type 3 DP:

CR cardinality

- one-to-one:
 - In a one-to-one communication relationship a CREP communicates with exactly one remote CREP.
- one-to-many:
 - In a one-to-many relationship one CREP communicates simultaneously with the CREPs of a group of devices. A one-to-many communication relationship does not permit confirmation of the execution of a request.

Connection mode:**connection-oriented**

In a connection-oriented application relationship a logical connection is established between application processes. Three different phases are identified in connection-oriented ARs:

- connection establishment phase
- data transfer phase and
- connection release phase

Exchanging data on a connection shall only be possible after a successful connection establishment. In the connection establishment phase a request to establish the connection is announced at the remote application process (e.g. Initiate service). The request to establish the connection-oriented AR contains the information about the context to be used in the data transfer phase. The remote application informs the requesting application process about the acceptance of the context. If the context is accepted both application processes are in the data transfer phase and are able to communicate with each other according to the agreed context. Connection-oriented AR can be monitored. A connection-oriented AR is dissolved by its release (e.g. Abort service).

connectionless

Connectionless application relationships have neither a connection establishment nor a connection release. They are always in the data transfer phase. It is not possible to monitor the connectionless AR.

Transmission types:**cyclic**

The transmission type "cyclic" means that after an initial request by the application process the application relationship is established and afterwards the data transfer is executed periodically.

acyclic

The transmission type "acyclic" means that a single transfer is executed after each request by the application process.

both

The transmission type "both" means that a cyclic as well as acyclic transfer is supported.

Transmission characteristics**Buffer to Buffer**

The transmission is characterised by a buffer to buffer transfer.

Queue to Queue

The transmission is characterised by a queue to queue transfer.

Buffer to Queue

The transmission is characterised by a buffer to queue transfer.

Queue to Buffer

The transmission is characterised by a queue to buffer transfer.

Service types

Confirmed

Only confirmed services are allowed.

Unconfirmed

Only unconfirmed services are allowed.

Transmission directions:

Bi-directional

APDUs are transmitted in both direction.

Unidirectional

APDUs are transmitted only in one direction.

9.2.7.1.1.4 Overview of Communication Relationships

The DP Application Layer uses the following types of communication relationships:

BBUB (Buffer Buffer Unconfirmed Bidirectional)

This type of communication relationship is characterised by the following role attributes:

- one to one
- connection-oriented
- cyclic
- buffer to buffer
- unconfirmed
- bidirectional

BQUU (Buffer Queue Unconfirmed Unidirectional)

This type of communication relationship is characterised by the following role attributes:

- one to one
- connectionless
- acyclic
- buffer to queue
- unconfirmed
- unidirectional

QQUU-OM (Queue Queue Unconfirmed Unidirectional - One to Many)

This type of communication relationship is characterised by the following role attributes:

- one to many
- connectionless
- acyclic
- queue to queue
- unconfirmed
- unidirectional

BBUU (Buffer Buffer Unconfirmed Unidirectional)

This type of communication relationship is characterised by the following role attributes:

- one to one
- connectionless
- acyclic
- buffer to buffer
- unconfirmed
- unidirectional

QQCB-CO (Queue Queue Confirmed Bidirectional - Connection-Oriented)

This type of communication relationship is characterised by the following role attributes:

- one to one
- connection-oriented
- acyclic
- queue to queue
- confirmed
- bidirectional

QQUU-OO (Queue Queue Unconfirmed Unidirectional - One to One)

This type of communication relationship is characterised by the following role attributes:

- one to one
- connectionless
- acyclic
- queue to queue
- unconfirmed
- unidirectional

QBCB (Queue Buffer Confirmed Bidirectional)

This type of communication relationship is characterised by the following role attributes:

- one to one
- connectionless
- acyclic
- queue to buffer
- confirmed
- bidirectional

QQCB-CL (Queue Queue Confirmed Bidirectional - Connectionless)

This type of communication relationship is characterised by the following role attributes:

- one to one
- connectionless
- acyclic
- queue to queue
- confirmed
- bidirectional

9.3 Type 3 ARs

The DP Application Layer offers a set of AR types.

9.3.1 MS0 Application Relationship

The MS0 AR reflects the application relationship (see Figure 52) between the application processes of one DP-Master (Class 1) and optional the AP of one or several DP-Master (Class 2) with all related DP-Slaves for the following purposes:

- cyclic exchange of the I/O data (DP-Master (Class 1))
- acyclic data transfer for parameterisation, configuration and diagnosis (DP-Master (Class 1))
- acyclic transfer of commands to a set of field devices (DP-Master (Class 1))
- acyclic read of the I/O data (DP-Master (Class 2))
- acyclic read of configuration information (DP-Master (Class 2))
- acyclic read of diagnosis information (DP-Master (Class 2))
- acyclic write of remanent parameter (DP-Master (Class 2))

This type of application relationship is characterised by the following role attributes:

- many to many
- Master-Slave communication
- Client Server relationship

The MS0 AR contains the following CR types:

- BBUB (Data Exchange)
- BQUU (Parameterisation and Configuration)
- BBUU (Slave Diag)
- QQUU-OM (Global Control)
- QBCB (Slave Diag, Get Cfg, Read Input, Read Output, Set Slave Add)

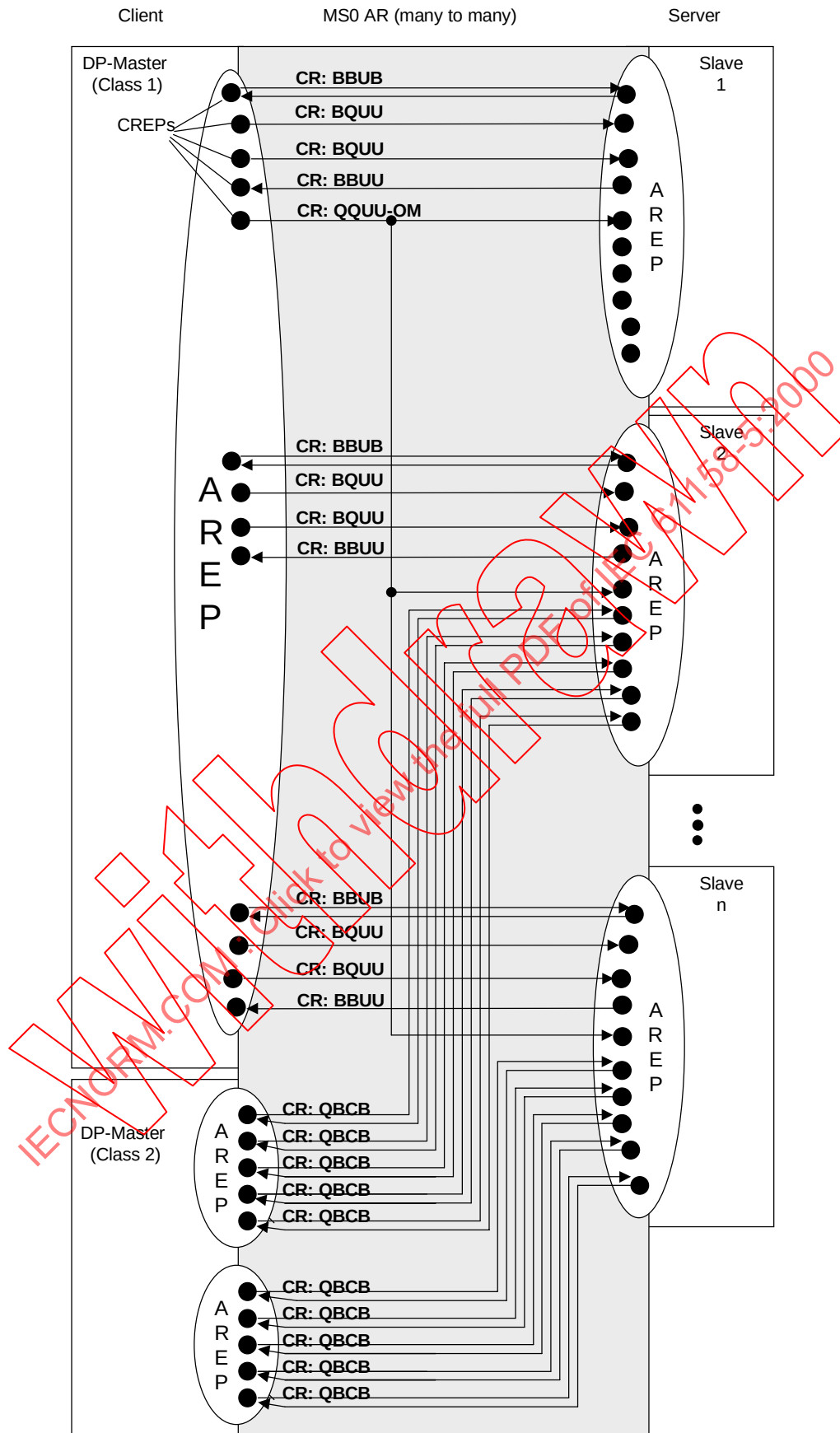


Figure 52 – MS0 Application Relationship

The DP-Application Layer incorporates buffers to convey the input and output data. This buffers decouple the communication and data processing. For synchronisation purposes two buffers for the Input Data and two buffers for the Outputs are necessary in the DP-Application Layer. If a DP-Slave supports no synchronisation capabilities only one buffer is needed in each direction. The detailed model is described in the following clauses.

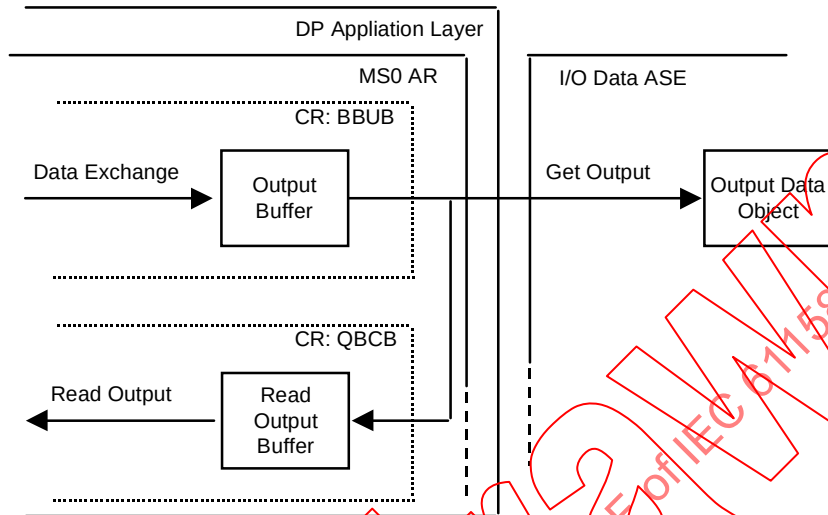


Figure 53 – Output Buffer model of a DP-Slave without Sync functionality

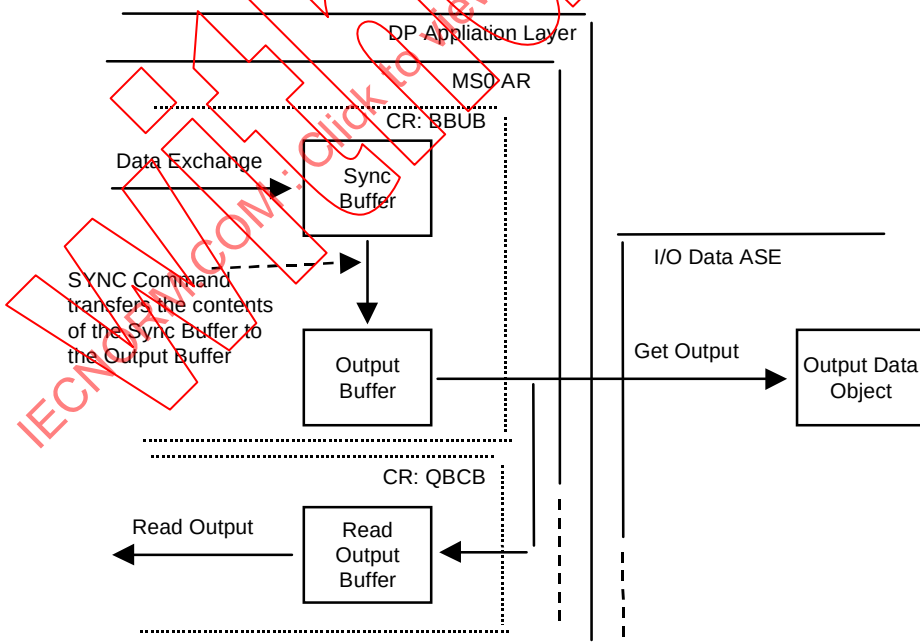


Figure 54 – Output Buffer model of a DP-Slave with Sync functionality

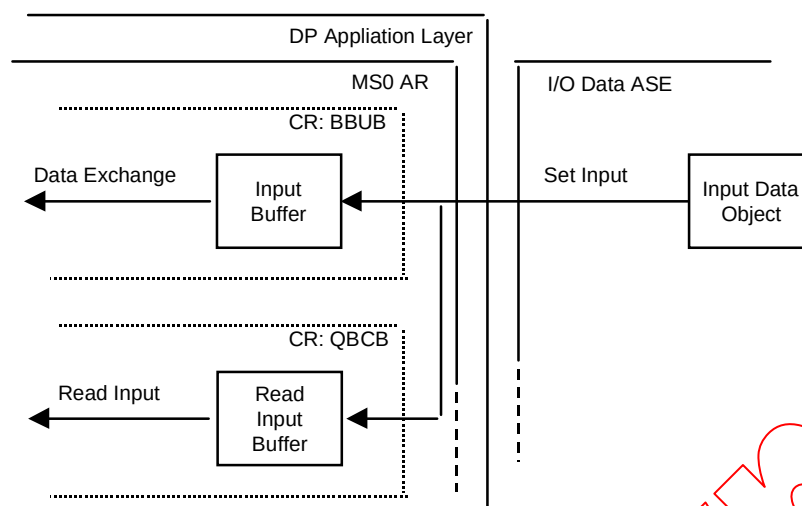


Figure 55 – Input Buffer model of a DP-Slave without Freeze functionality

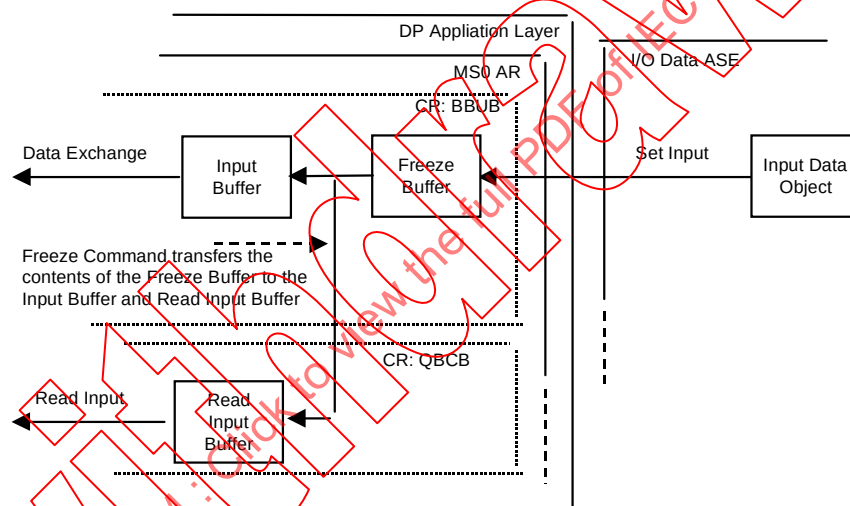


Figure 56 – Input Buffer model of a DP-Slave with Freeze functionality

9.3.2 MS1 Application Relationship

The MS1 AR reflects the connection-oriented application relationship (see Figure 57) between the application processes of one DP-Master (Class 1) and of one related DP-Slave for the following purposes:

- acyclic read and write of variables
- acyclic transfer of alarms

This type of application relationship is characterised by the following role attributes:

- one to one
- Master-Slave communication
- Client Server relationship

The MS1 AR contains the following CR types:

- QQCB-CO (Process Data, Alarm Ack)
- QQUU-OO (Alarm)

and optionally the following CR type:

- QQCB-CO (Alarm Ack)

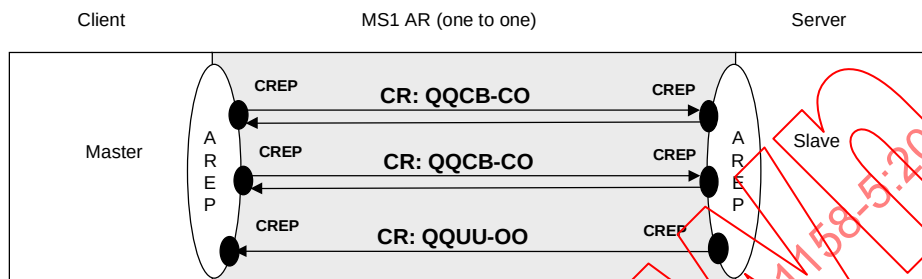


Figure 57 – MS1 Application Relationship

9.3.3 MS2 Application Relationship

The MS2 AR reflects the connection-oriented application relationship (see Figure 58) between the application processes of one DP-Master (Class 2) and of one related DP-Slave for the purpose:

- acyclic read and write of variables

This type of application relationship is characterised by the following role attributes:

- one to one
- Master-Slave communication
- Client Server relationship

The MS2 AR contains the following CR type:

- QQCB-CO (Process Data)
- QQUU-OO (Resource Manager)

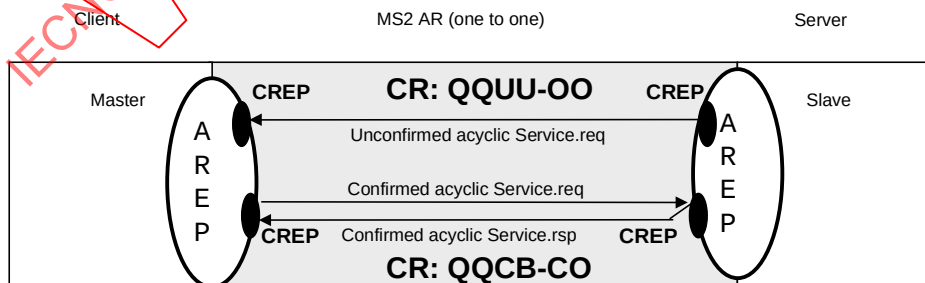


Figure 58 – MS2 Application Relationship

9.3.3.1 Communication over multiple interconnected networks

With the Initiate service it is possible to communicate over multiple networks.

Following cases are considered:

- data transfer from one DP-Network to another DP-Network over a link
- data transfer from a None-DP-Network to a DP-Network over a gateway

Within such networks each DP-Device including their applications are addressed with the following parameters:

- Network address
- MAC
- API, SCL

The format of these addresses is defined in the Initiate service description. It is necessary to distinguish between DP-Devices which are either used as a link or a communication endpoint. In the case of the link the Initiate-PDU has to be passed to the addressed device. The Network address 0 is reserved for default addressing purposes and therefore shall not be used as a normal network address.

The Figure 59 shows a cut of an heterogeneous network. Type 3 DP covers only the marked area.

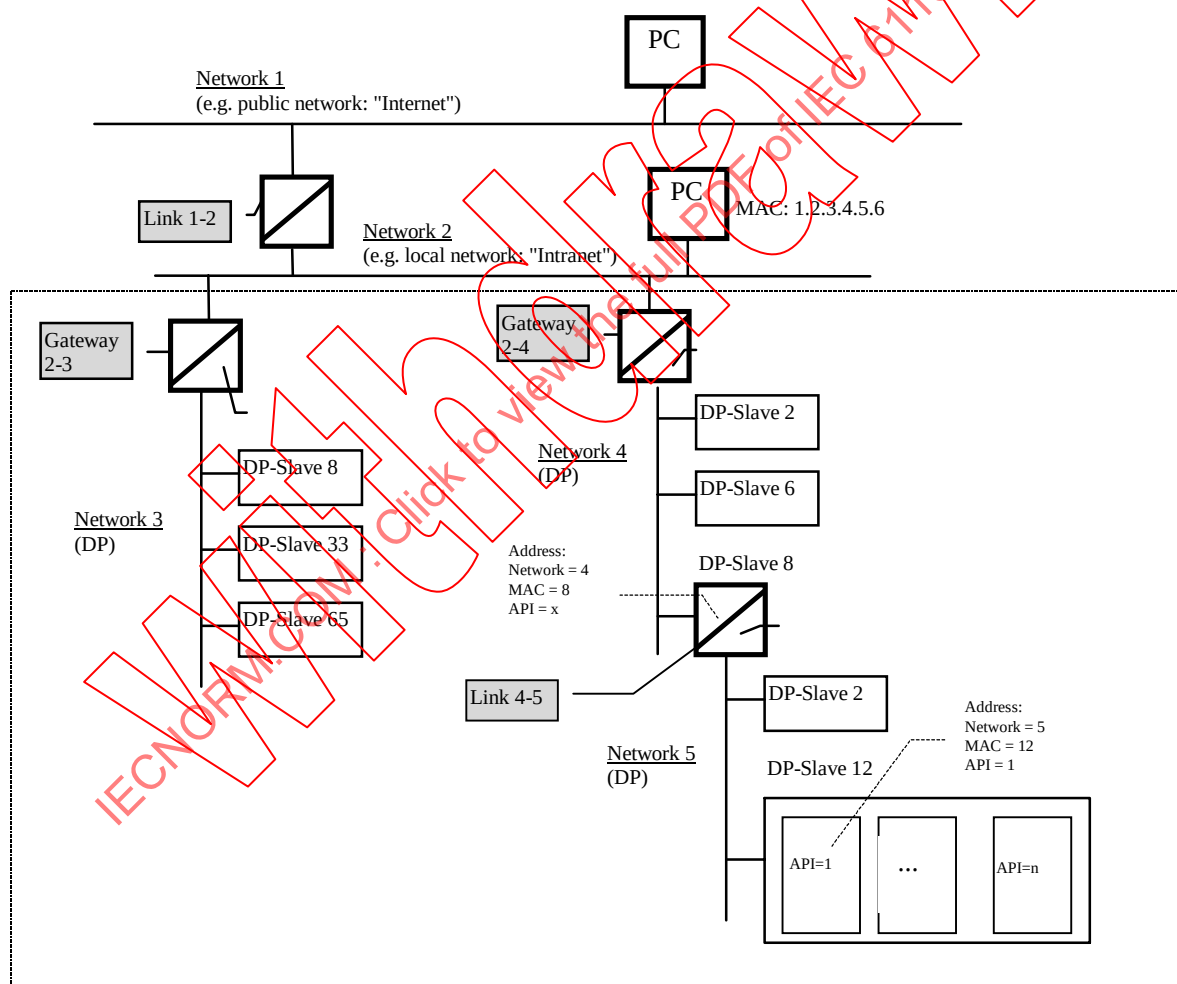


Figure 59 – Example of Inter-Network Communication

The following examples (see Figure 60 to Figure 62) shows the exchange of the source- and destination-address parameter of the Initiate.req and the Initiate.res service.

Between the AP 10 of DP-Master (Class 2) and AP 1 of the DP-Slave a communication relationship will be established. This is done with the Initiate service. After the connection establishment the transfer of data blocks will be realised with the services Read and Write using the parameters Slot Number and Index.

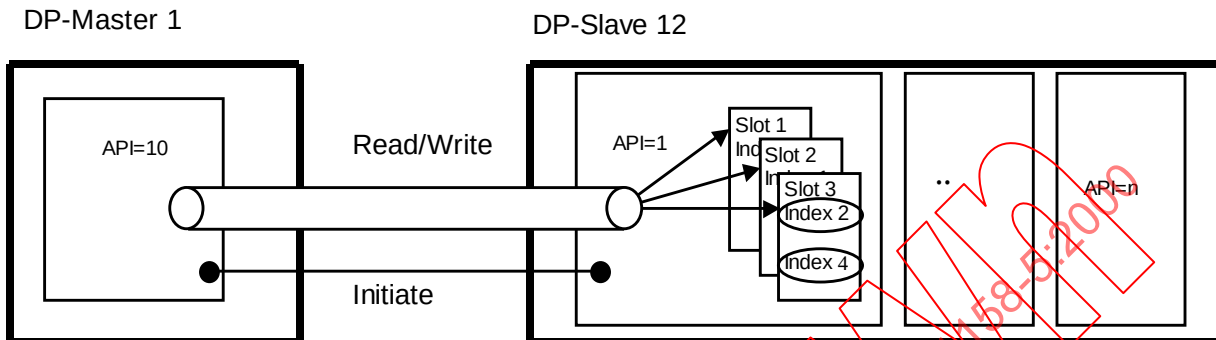


Figure 60 – Example without Inter-Network Addressing

Table 227 – Parameter of Initiate service without Inter-Network Addressing

Initiate.req (1 -> 12)		Initiate.res (12 -> 1)	
S_Type	0	S_Type	0
S_Len	2	S_Len	2
D_Type	0	D_Type	0
D_Len	2	D_Len	2
S_Addr	API 10, SCL	S_Addr	API 1, SCL
D_Addr	API 1, SCL	D_Addr	API 10, SCL

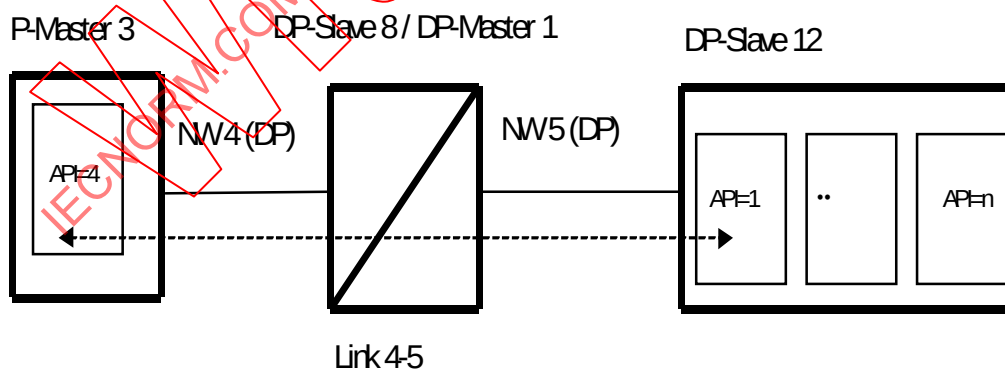


Figure 61 – First Example with Inter-Network Addressing

Table 228 – Parameter of Initiate service with Inter-Network Addressing (first example)

Initiate.req (3 -> 8)		Initiate.res (8 -> 3)	
S_Type	0	S_Type	1
S_Len	2	S_Len	9
D_Type	1	D_Type	0
D_Len	9	D_Len	2
S_Addr	API 4, SCL	S_Addr	API 1, SCL, NW5, 12
D_Addr	API 1, SCL, NW5, 12	D_Addr	API 4, SCL
Initiate.req (1 -> 12)		Initiate.res (12 -> 1)	
S_Type	1	S_Type	0
S_Len	9	S_Len	2
D_Type	0	D_Type	1
D_Len	2	D_Len	9
S_Addr	API 4, SCL, NW4, 3	S_Addr	API 1, SCL
D_Addr	API 1, SCL	D_Addr	API 4, SCL, NW4, 3

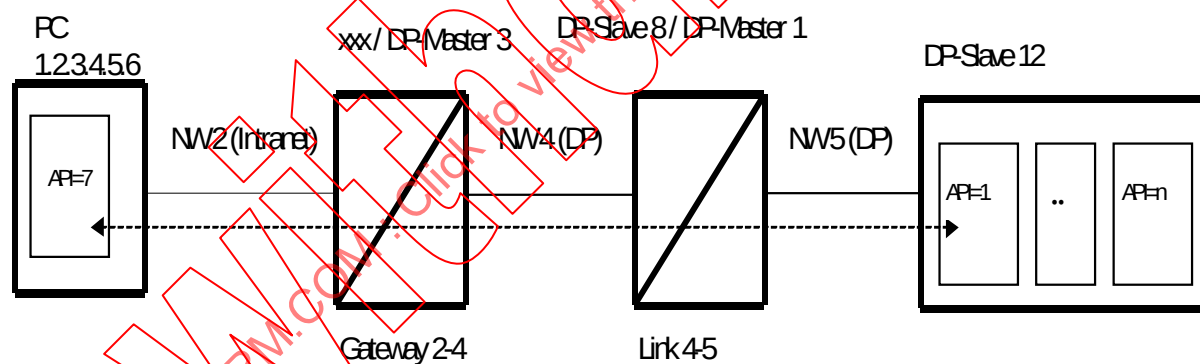
**Figure 62 – Second Example with Inter-Network Addressing**

Table 229 – Parameter of Initiate service with Inter-Network Addressing (second example)

Initiate-REQ-PDU (3 -> 8)		Initiate-RES-PDU (8 -> 3)	
S_Type	1	S_Type	1
S_Len	14	S_Len	9
D_Type	1	D_Type	1
D_Len	9	D_Len	14
S_Addr	API 7, SCL, NW2, 1.2.3.4.5.6	S_Addr	API 1, SCL, NW5, 12
D_Addr	API 1, SCL, NW5, 12	D_Addr	API 7, SCL, NW2, 1.2.3.4.5.6
Initiate-REQ-PDU (1 -> 12)		Initiate-RES-PDU (12 -> 1)	
S_Type	1	S_Type	0
S_Len	14	S_Len	2
D_Type	0	D_Type	1
D_Len	2	D_Len	14
S_Addr	API 7, SCL, NW2, 1.2.3.4.5.6	S_Addr	API 1, SCL
D_Addr	API 1, SCL	D_Addr	API 7, SCL, NW2, 1.2.3.4.5.6

9.3.4 MM1 Application Relationship

The MM1 AR reflects the connectionless relationship (see Figure 63) between the application processes of one configuration device (Master Class 2) and of one related controlling device (Master Class 1) for the following purposes:

- upload and download for configuration information
- upload of diagnosis information
- activation of the previously transferred configuration

This type of application relationship is characterised by the following role attributes:

- one to one
- Master-Master communication
- Client Server relationship

The MM1 AR contains the following CR type:

- QQCB-CL (Master Diag/Start Seq/End Seq/Download/Upload/Act Param)

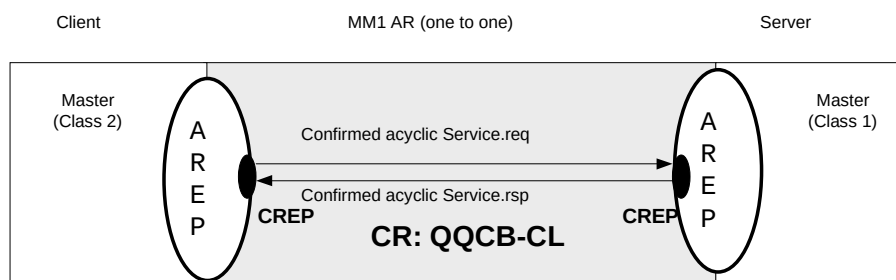


Figure 63 – MM1 Application Relationship

9.3.5 MM2 Application Relationship

The MM2 AR reflects the connectionless relationship (see Figure 64) between the application processes of one configuration device (Master Class 2) and of a set of related controlling devices (Master Class 1) for the purpose:

- activation of the previously transferred configurations

This type of application relationship is characterised by the following role attributes:

- one to many
- Master-Master communication
- Client Server relationship

The MM2 AR contains the following CR type:

- QQUU-OM (Act Para Brct)

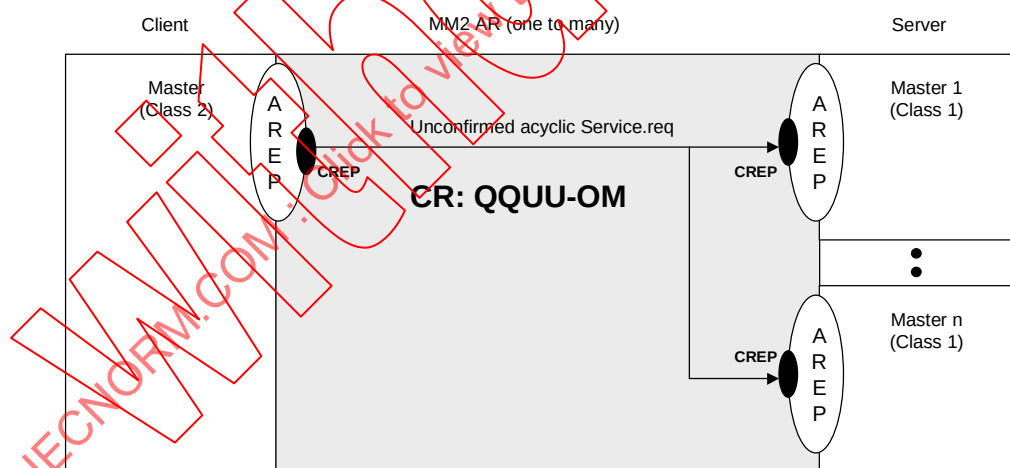


Figure 64 – MM2 Application Relationship

9.3.5.1 Application Relationship List Class Specification

The ARL Class specification is distinguished between DP-Slave, DP-Master (Class 1) and DP-Master (Class 2).

A static attribute is assigned by a load service and are not changed during operation, whereas a dynamic attribute may set by the load services and may be changed during operation.

9.3.5.1.1 ARL DP-Slave Class Specification

The ARL DP-Slave object is described by the following template:

DP ASE:	AR ASE
CLASS:	ARL DP-Slave
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m)	Key Attribute: Implicit
2. (c)	Constraint: at least one entry with AR Type = MS2
2.1. (m)	Attribute: Min Send Timeout
2.2. (m)	Attribute: List of Resource Manager Entries
2.2.1. (m)	Attribute: API
2.2.2. (m)	Attribute: SCL
2.2.3. (m)	Attribute: Max Len Data Unit
2.2.4. (m)	Attribute: Send Timeout
2.2.5. (m)	Attribute: AREP
2.2.6. (m)	Attribute: S_SAP_index
2.2.7. (m)	Attribute: In Use
2.3. (m)	Attribute: List of S_SAP_indices
2.3.1. (m)	Attribute: S_SAP_index
3. (m)	Attribute: List of ARL Entries
3.1. (m)	Attribute: AREP
3.2. (m)	Attribute: AR Type
3.3. (m)	Attribute: List of related CRL Entries
3.3.1. (m)	Attribute: CREP
3.4. (c)	Constraint: AR Type = MS0
3.4.1. (m)	Attribute: Prm Req
3.4.2. (m)	Attribute: Prm Fault
3.4.3. (m)	Attribute: Cfg Fault
3.4.4. (m)	Attribute: Not Supported
3.4.5. (m)	Attribute: Station Not Ready
3.4.6. (m)	Attribute: Stat Diag
3.4.7. (m)	Attribute: Master Add
3.4.8. (m)	Attribute: Sync Supported
3.4.9. (m)	Attribute: Sync Mode
3.4.10. (m)	Attribute: Freeze Supported
3.4.11. (m)	Attribute: Freeze Mode
3.4.12. (m)	Attribute: WD Enabled
3.4.13. (m)	Attribute: WD Fact 1
3.4.14. (m)	Attribute: WD Fact 2
3.4.15. (m)	Attribute: Min TSDR
3.4.16. (m)	Attribute: Ident Number

3.4.17. (m)	Attribute:	Group Identifier
3.4.18. (m)	Attribute:	DPV1 Supported
3.4.19. (c)	Constraint:	DPV1 Supported = TRUE
3.4.19.1. (m)	Attribute:	DPV1 Enabled
3.4.19.2. (m)	Attribute:	Fail Safe Required
3.4.19.3. (m)	Attribute:	Fail Safe
3.4.19.4. (m)	Attribute:	WD Base
3.4.20. (m)	Attribute:	List of Real Cfg Data
3.4.20.1. (s)	Attribute:	Cfg Identifier
3.4.20.1.1. (m)	Attribute:	Type
3.4.20.1.2. (m)	Attribute:	Format
3.4.20.1.3. (m)	Attribute:	Consistency
3.4.20.1.4. (m)	Attribute:	Length
3.4.20.2. (s)	Attribute:	Special Cfg Identifier
3.4.20.2.1. (m)	Attribute:	Type
3.4.20.2.2. (c)	Constraint:	Type = Input or Type = Input-Output
3.4.20.2.2.1. (m)	Attribute:	Input Length
3.4.20.2.2.2. (m)	Attribute:	Input Format
3.4.20.2.2.3. (m)	Attribute:	Input Consistency
3.4.20.2.3. (c)	Constraint:	Type = Output or Type = Input-Output
3.4.20.2.3.1. (m)	Attribute:	Output Length
3.4.20.2.3.2. (m)	Attribute:	Output Format
3.4.20.2.3.3. (m)	Attribute:	Output Consistency
3.4.20.2.4. (o)	Attribute:	List of Data Description
3.4.20.2.4.1. (m)	Attribute:	Data Description
3.4.20.2.5. (o)	Attribute:	Manufacturer Specific Data
3.4.21. (o)	Attribute:	No Add Change
3.5. (c)	Constraint:	AR Type = MS1
3.5.1. (m)	Attribute:	Pull Plug Alarm supported
3.5.2. (c)	Constraint:	Pull Plug Alarm supported = TRUE
3.5.2.1. (m)	Attribute:	Pull Plug Alarm enabled
3.5.3. (m)	Attribute:	Process Alarm supported
3.5.4. (c)	Constraint:	Process Alarm supported = TRUE
3.5.4.1. (m)	Attribute:	Process Alarm enabled
3.5.5. (m)	Attribute:	Diagnosis Alarm supported
3.5.6. (c)	Constraint:	Diagnosis Alarm supported = TRUE
3.5.6.1. (m)	Attribute:	Diagnosis Alarm enabled
3.5.7. (m)	Attribute:	Manufacturer Specific Alarm supported
3.5.8. (c)	Constraint:	Manufacturer Specific Alarm supported = TRUE
3.5.8.1. (m)	Attribute:	Manufacturer Specific Alarm enabled
3.5.9. (m)	Attribute:	Status Alarm supported
3.5.10. (c)	Constraint:	Status Alarm supported = TRUE

- 3.5.10.1. (m) Attribute: Status Alarm enabled
- 3.5.11. (m) Attribute: Update Alarm supported
- 3.5.12. (c) Constraint: Update Alarm supported = TRUE
- 3.5.12.1. (m) Attribute: Update Alarm enabled
- 3.5.13. (m) Attribute: Alarm Mode supported
- 3.5.14. (c) Constraint: Alarm Mode supported = TRUE
- 3.5.14.1. (m) Attribute: Alarm Mode enabled
- 3.6. (o) Attribute: AR Name

SERVICES:

- 1. (m) MgtService: Load ARL
- 2. (o) MgtService: Get ARL
- 3. (m) OpsService: Set Cfg

9.3.5.1.1.1 Attributes

Implicit

The attribute Implicit indicates that the ARL DP-Slave object is implicitly addressed by the service.

Min Send Timeout

This conditional and static attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. This attribute defines the minimum Send Timeout common for all MS2 ARs supported in this station.

Allowed value: 0 to $2^{16}-1$

Time Base: 10ms

List of Resource Manager Entries

This conditional attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. This attribute is common for all MS2 ARs supported in this station and composed of the following list elements:

API

This conditional and static attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. The attribute API specifies the numeric Identifier for the related Application Process of this MS2 AR.

Allowed values: 0 to 255 (0 = Default AP)

SCL

This conditional and static attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. The attribute SCL specifies the configured Security Level for the identified Application Process of this MS2 AR.

Allowed values: 0 to 255 (0 = no Security Level used)

Max Len Data Unit

This conditional and static attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. This attribute reflects the upper limit of the parameter Length of the services of the Process Data ASE for this MS2 AR.

Allowed values: 0 to 240

Send Timeout

This conditional and dynamic attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. This attribute defines the Send Timeout for this MS2 AR. This attribute is set to the maximum of the value of the attribute Min Send Timeout and the field Send Timeout of the previously received Initiate-REQ-PDU for this MS2 AR. This attribute is not influenced by the Load ARL service.

Allowed value: 0 to $2^{16}-1$

Time Base: 10ms

AREP

This conditional and static attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. This attribute specifies the local identifier of this MS2 AR.

S_SAP_index

This conditional and dynamic attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. This attribute contains the assigned S_SAP_index from the list of S_SAP_indices after reception of an Initiate-REQ-PDU for this MS2 AR. This attribute is not influenced by the Load ARL service.

Allowed values: 0 to 48

In Use

This conditional and dynamic attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. This attribute specifies the state of this MS2 AR. This attribute is set to TRUE after reception of an Initiate-REQ-PDU and assignment of a S_SAP_index for this MS2 AR. This attribute is set to FALSE after release of this MS2 AR. This attribute is not influenced by the Load ARL service.

Allowed values: FALSE, TRUE

List of S_SAP_indices

This conditional attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. This attribute is common for all MS2 ARs supported in this station and composed of the following list elements:

S_SAP_index

This conditional and static attribute shall only present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS2. This attribute specifies an available Responder SAP for the MS2 ARs of the DP-Slave.

Allowed values: 0 to 48

List of ARL Entries

This attribute is composed of the following list elements:

AREP

This static attribute contains the local identifier of this AR.

AR Type

This static attribute specifies the type of this AR.

Allowed values: MS0, MS1, MS2

List of related CRL Entries

This attribute is composed of the following list elements:

CREP

This static attribute contains the local identifier for the related CR for this AR.

AR Type = MS0

This constraint reflects the attributes for this AR if the value of the attribute AR Type is set to MS0.

Prm Req

This dynamic attribute indicates if the DP-Slave has to be reparameterised and reconfigured. This attribute shall be set to TRUE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: TRUE, FALSE

Prm Fault

This dynamic attribute indicates the result of the check of the parameterisation. This attribute shall be set to FALSE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: TRUE, FALSE

Cfg Fault

This dynamic attribute indicates the result of the check of the configuration data. This attribute shall be set to FALSE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: TRUE, FALSE

Not Supported

This dynamic attribute indicates that a functionality requested by the DP-Master (Class 1) is not supported on the MS0 AR by the DP-Slave. This attribute shall be set to FALSE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: TRUE, FALSE

Station Not Ready

This dynamic attribute indicates the readiness to enter the data exchange mode with the assigned DP-Master (Class 1). This attribute shall be set to TRUE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: TRUE, FALSE

Stat Diag

This dynamic attribute indicates that the DP-Master (Class 1) shall fetch permanently diagnosis from the DP-Slave until the application indicates readiness for operation with the Application Ready service. This attribute shall be set to FALSE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine and the Application Ready service.

Allowed values: TRUE, FALSE

Master Add

This dynamic attribute contains the DL address of the DP-Master (Class 1) which has locked the DP-Slave. This attribute shall be set to invalid (= 255) with the Load ARL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: 0 to 126, 255

Sync Supported

This static attribute reflects the ability of a DP-Slave to participate in the synchronisation of the Output Data of all or a group of DP-Slaves. As the check whether the Sync ability is supported is done during parameterisation phase, errors will be avoided during the data exchange mode. This attribute shall be set accordingly with the Load ARL service. Table 230 shows the allowed values for this attribute.

Table 230 – Sync Supported

Value	Meaning
FALSE	The DP-Slave does not support the Sync ability
TRUE	The DP-Slave supports the Sync ability

Sync Mode

This dynamic attribute indicates if the Sync mode is activated. This attribute shall always be set to FALSE if the attribute Sync Supported has the value FALSE. If the attribute Sync Supported has the value TRUE the following rules shall apply for the attribute Sync Mode:

It shall be set to TRUE as soon as the control command Sync is received, which is conveyed by the Global Control service.

It shall be set to FALSE as soon as the control command Unsync is received which is conveyed by the Global Control service.

This attribute shall be set to FALSE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine.

Freeze Supported

This static attribute reflects the ability of a DP-Slave to participate in the synchronisation of the Input Data of all or a group of DP-Slaves. As the check whether the Freeze ability is supported is done during parameterisation phase, errors will be avoided during the data exchange mode. This attribute shall be set accordingly with the Load ARL service. Table 231 shows the allowed values for this attribute.

Table 231 – Freeze Supported

Value	Meaning
FALSE	The DP-Slave does not support the Freeze ability
TRUE	The DP-Slave supports the Freeze ability

Freeze Mode

This dynamic attribute indicates if the Freeze mode is activated. This attribute shall always be set to FALSE if the attribute Freeze Supported has the value FALSE. If the attribute Freeze Supported has the value TRUE the following rules shall apply for the attribute Freeze Mode:

It shall be set to TRUE as soon as the control command Freeze is received, which is conveyed by the Global Control service.

It shall be set to FALSE as soon as the control command Unfreeze is received which is conveyed by the Global Control service.

This attribute shall be set to FALSE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine.

WD Enabled

This dynamic attribute indicates if the watchdog control is activated. This attribute shall be set to FALSE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: TRUE, FALSE

WD Fact 1, WD Fact 2

These dynamic attributes contain the values for the monitoring of the DP-Master (Class 1) at the MS0 AR (watchdog control (T_{WD})). The watchdog control in a DP-Slave takes care that, if the DP-Master fails, the outputs will be set in the safe state after the expiration of this time. The time is calculated according to equation:

$$T_{WD} = \text{Watchdog Time Base} \times \text{WD Fact 1} \times \text{WD Fact 2}$$

If the attribute DPV1 Supported has the value FALSE the Watchdog Time Base shall be set to 10 ms.

If the attribute DPV1 Supported has the value TRUE the Watchdog Time Base shall be set to the value 1ms if the attribute WD Base has the value TRUE. If the attribute DPV1 Supported has the value TRUE the Watchdog Time Base shall be set to the value 10ms if the attribute WD Base has the value FALSE. These attributes are influenced by the MS0 AR state machine. These attributes are not influenced by the Load ARL service.

Allowed values: 1 to 255

Min TSDR

This dynamic attribute contains the value for the Minimum Station Delay Responder time of the DLL. If this Parameter is equal zero, the previous value of the Min TSDR remains unchanged. This attribute is influenced by the MS0 AR state machine. This attribute is not influenced by the Load ARL service.

Allowed values: 0 to 255

Ident Number

This static attribute contains the Ident Number of the DP-Slave. The DP-Slave accepts the parameterisation and change of the DL address only if the transmitted Ident Number in the correspondent service is identical with this attribute. This attribute shall be set accordingly with the Load ARL service.

Allowed values: 0 to $2^{16}-1$

Group Identifier

This dynamic attribute indicates the group(s) to which this DP-Slave has been assigned by the DP-Master (Class 1) in the parameterisation. The Group Identifier distinguishes 8 groups. If the Group Identifier is set to No Group, the DP-Slave is not assigned to any group. This attribute is influenced by the MS0 AR state machine. This attribute is not influenced by the Load ARL service.

Allowed values: Group 1 to Group 8, No Group and combinations

DPV1 Supported

This static attribute indicates whether the DP-Slave supports the extended functionality. This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FALSE

DPV1 Enabled

This dynamic attribute indicates if the extended functionality is activated. This conditional attribute shall only be present if the attribute DPV1 Supported has the value TRUE. The following rules shall apply for the attribute DPV1 Enabled:

It shall be set to TRUE if the DP-Master (Class 1) requests the DPV1 support in the parameterisation.

It shall be set to FALSE if the DP-Master (Class 1) requests no DPV1 support in the parameterisation.

This attribute shall be set to FALSE with the Load ARL service. This attribute is further influenced by the MS0 AR state machine. Table 232 shows the allowed values for this attribute.

Table 232 – DPV1 Enabled

Value	Meaning
FALSE	The extended functionality related with the attribute DPV1 Supported is disabled.
TRUE	The extended functionality related with the attribute DPV1 Supported is enabled.

Fail Safe Required

This static attribute indicates if the DP-Slave requires the Fail Safe mode for operation. This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FALSE

Fail Safe

This dynamic attribute indicates if the DP-Master (Class 1) supports the Fail Safe mode. If the DP-Slave requires the Fail Safe mode and the DP-Master (Class 1) does not support this mode the DP-Slave shall respond with an error entry in the Slave Diagnosis (Prm Fault). This attribute is influenced by the MS0 AR state machine. This attribute is not influenced by the Load ARL service. Table 233 shows the allowed values for this attribute.

Table 233 – Fail Safe

Value	Meaning
FALSE	The DP-Master (Class 1) does not support the Fail Safe mode.
TRUE	The DP-Master (Class 1) supports the Fail Safe mode.

WD Base

This dynamic attribute contains the Time Base value for the watchdog control according to the parameterisation. This attribute is influenced by the MS0 AR state machine. This attribute is not influenced by the Load ARL service. Table 234 shows the allowed values for this attribute.

Table 234 – WD Base

Value	Meaning
FALSE	The Watchdog Time Base is 10ms.
TRUE	The Watchdog Time Base is 1ms.

List of Real Cfg Data

This dynamic attribute is composed of the following list elements:

Cfg Identifier

This dynamic attribute describes the basic configuration format and is composed of the following elements: This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Type

This dynamic attribute indicates the type of the related Input/Output Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: Input, Output, Input-Output

Format

This dynamic attribute indicates the format of the related Input/Output Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: Byte, Word

Consistency

This dynamic attribute indicates the consistency of the related Input/Output Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: Byte, Word, whole length

Length

This dynamic attribute indicates the length of the related Input/Output Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: 1 to 32

Special Cfg Identifier

This dynamic attribute describes the special configuration format and is composed of the following elements. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Type

This dynamic attribute indicates the type of the related Input/Output Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: Input, Output, Input-Output

Input Length

This dynamic attribute indicates the length of the related Input Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: 1 to 128

Input Format

This dynamic attribute indicates the format of the related Input Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: Byte, Word

Input Consistency

This dynamic attribute indicates the consistency of the related Input Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: Byte, Word, whole length

Output Length

This dynamic attribute indicates the length of the related Output Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: 1 to 128

Output Format

This dynamic attribute indicates the format of the related Output Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: Byte, Word

Output Consistency

This dynamic attribute indicates the consistency of the related Output Data Element. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Allowed values: Byte, Word, whole length

List of Data Description

This dynamic attribute is composed of the following list elements. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Data Description

This dynamic attribute contains one of the allowed Data Types of the data description of one element of the Extended Input/Output Data object. The allowed Data Types are given in 5.2.1. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

Manufacturer Specific Data

This dynamic attribute contains the manufacturer specific data for one Special Cfg Identifier. This attribute shall be set accordingly with the Load ARL service. Furthermore the attribute is influenced by the Set Cfg service.

No Add Change

This dynamic attribute indicates whether it is allowed to change the DP-Slave address. To change this attribute itself an initial reset of the DP-Slave is necessary. After the initial reset the DP-Slave takes the default address 126 and the attribute No Add Change will be set to FALSE. This attribute shall be set accordingly with the Load ARL service. This attribute is further influenced by the MS0 AR state machine. Table 235 shows the allowed values for this attribute.

Table 235 – No Add Change

Value	Meaning
FALSE	Change of the address is allowed
TRUE	Change of the address is not allowed

AR Type = MS1

This constraint reflects the attributes for this AR if the value of the attribute AR Type is set to MS1. Furthermore this attribute shall only be present if the attribute DPV1 Supported is set to TRUE in the MS0 AR entry.

Pull Plug Alarm supported

This static attribute indicates the capability of a DP-Slave to send a Pull or a Plug Alarm. This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FALSE

Pull Plug Alarm enabled

This dynamic attribute indicates if the transmission of the Pull Plug Alarm is enabled by the DP-Master (Class 1). This conditional attribute shall only be present if the attribute Pull Plug Alarm supported is set to TRUE. This attribute is influenced by the MS1 AR state machine. This attribute is not influenced by the Load ARL service.

Allowed values: TRUE, FALSE

Process Alarm supported

This static attribute indicates the capability of a DP-Slave to send a Process Alarm. This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FALSE

Process Alarm enabled

This dynamic attribute indicates if the transmission of the Process Alarm is enabled by the DP-Master (Class 1). This conditional attribute shall only be present if the attribute Process Alarm supported is set to TRUE. This attribute is influenced by the MS1 AR state machine. This attribute is not influenced by the Load ARL service.

Allowed values: TRUE, FALSE

Diagnosis Alarm supported

This static attribute indicates the capability of a DP-Slave to send a Diagnosis Alarm. This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FALSE

Diagnosis Alarm enabled

This dynamic attribute indicates if the transmission of the Diagnosis Alarm is enabled by the DP-Master (Class 1). This conditional attribute shall only be present if the attribute Diagnosis Alarm supported is set to TRUE. This attribute is influenced by the MS1 AR state machine. This attribute is not influenced by the Load ARL service.

Allowed values: TRUE, FALSE

Manufacturer Specific Alarm supported

This static attribute indicates the capability of a DP-Slave to send a Manufacturer Specific Alarm. This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FALSE

Manufacturer Specific Alarm enabled

This dynamic attribute indicates if the transmission of the Manufacturer Specific Alarm is enabled by the DP-Master (Class 1). This conditional attribute shall only be present if the attribute Manufacturer Specific Alarm supported is set to TRUE. This attribute is influenced by the MS1 AR state machine. This attribute is not influenced by the Load ARL service.

Allowed values: TRUE, FALSE

Status Alarm supported

This static attribute indicates the capability of a DP-Slave to send a Status Alarm. This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FALSE

Status Alarm enabled

This dynamic attribute indicates if the transmission of the Status Alarm is enabled by the DP-Master (Class 1). This conditional attribute shall only be present if the attribute Status Alarm supported is set to TRUE. This attribute is influenced by the MS1 AR state machine. This attribute is not influenced by the Load ARL service.

Allowed values: TRUE, FALSE

Update Alarm supported

This static attribute indicates the capability of a DP-Slave to send a Update Alarm. This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FALSE

Update Alarm enabled

This dynamic attribute indicates if the transmission of the Update Alarm is enabled by the DP-Master (Class 1). This conditional attribute shall only be present if the attribute Update Alarm supported is set to TRUE. This attribute is influenced by the MS1 AR state machine. This attribute is not influenced by the Load ARL service.

Allowed values: TRUE, FALSE

Alarm Mode supported

This static attribute indicates the maximum number of possible active alarms in a DP-Slave. This attribute shall be set accordingly with the Load ARL service. Table 236 shows the allowed values for this attribute.

Table 236 – Alarm Mode supported

Value	Meaning
0	1 alarm of each type
1	2 alarms in total
2	4 alarms in total
3	8 alarms in total
4	12 alarms in total
5	16 alarms in total
6	24 alarms in total
7	32 alarms in total

Alarm Mode enabled

This dynamic attribute indicates the maximum number of possible active alarms negotiated with the DP-Master (Class 1) on this MS1 AR. The negotiation shall be done according to the following rules:

If the attribute Alarm Mode is set to the value 0 and the desired Alarm Mode of the DP-Master (Class 1) contained in the parameterisation is set to 0 than this attribute shall be set to 0.

If the attribute Alarm Mode is set to the value 0 and the desired Alarm Mode of the DP-Master (Class 1) contained in the parameterisation is set to a value different from 0 than this attribute shall not be set.

If the attribute Alarm Mode is set to a value different from 0 and the desired Alarm Mode of the DP-Master (Class 1) contained in the parameterisation is set to 0 than this attribute shall be set to 0.

If the attribute Alarm Mode is set to a value different from 0 and the desired Alarm Mode of the DP-Master (Class 1) contained in the parameterisation is set to a value different from 0 than this attribute shall be set to the lesser value of both.

This attribute is influenced by the MS1 AR state machine. This attribute is not influenced by the Load ARL service.

Allowed values : see Table 236

AR Name

This static attribute contains the name of the related ARL Entry. This attribute shall be set accordingly with the Load ARL service.

9.3.5.1.1.2 Invocation of ARL DP-Slave object

For the invocation of the ARL DP-Slave object the following rules have to be considered:

- Only one ARL DP-Slave object shall be invoked in a DP-Slave.
- Only one MS0 Entry shall be contained in the List of ARL Entries
- Only one MS1 Entry shall be contained in the List of ARL Entries

9.3.5.1.2 ARL DP-Master (Class 1) Class Specification

The ARL DP-Master (Class 1) object is described by the following template.

DP ASE: **AR ASE**
CLASS: **ARL DP-Master (Class 1)**
CLASS ID: not used
PARENT CLASS: TOP

ATTRIBUTES:

1. (m) Key Attribute: Implicit
2. (o) Attribute: Alarm Max
3. (m) Attribute: List of ARL Entries
 - 3.1. (m) Attribute: AREP
 - 3.2. (m) Attribute: AR Type
 - 3.3. (m) Attribute: List of related CRL Entries
 - 3.3.1. (m) Attribute: CREP
 - 3.4. (c) Constraint: AR Type = MS0
 - 3.4.1. (m) Attribute: Error Action Flag
 - 3.4.2. (m) Attribute: Min Slave Interval
 - 3.4.3. (m) Attribute: Data Control Time
 - 3.4.4. (m) Attribute: Max User Global Control
 - 3.4.5. (m) Attribute: Max Diag Data Len
 - 3.5. (c) Constraint: AR Type = MS1
 - 3.5.2. (m) Attribute: Extra Alarm SAP
 - 3.5.3. (m) Attribute: Max Channel Data Length
 - 3.5.4. (m) Attribute: Alarm Mode
 - 3.5.5. (m) Attribute: MS1 Timeout
 - 3.6. (c) Constraint: AR Type = MM1
 - 3.6.1. (m) Attribute: Poll Timeout
 - 3.6.2. (m) Attribute: Ident Number
 - 3.7. (o) Attribute: AR Name

SERVICES:

1. (m) MgtService: Load ARL
2. (o) MgtService: Get ARL

9.3.5.1.2.1 Attributes

Implicit

The attribute Implicit indicates that the ARL DP-Slave object is implicitly addressed by the service.

Alarm Max

This conditional and static attribute shall only be present if the List of ARL Entries contains at least one entry with the attribute AR Type set to MS1. It indicates the maximum number of alarms per DP-Slave, which can be handled by the DP-Master (Class 1). It describes a Master capability and is common for all MS1 ARs. The Master shall be able to handle a minimum number of 7 alarms, because there are existing 7 different alarm types. This attribute shall be set accordingly with the Load ARL service.

Allowed values: 7 to 32

List of ARL Entries

This attribute is composed of the following list elements:

AREP

This static attribute contains the local identifier of this AR.

AR Type

This static attribute specifies the type of this AR.

Allowed values: MS0, MS1, MM1, MM2

List of related CRL Entries

This attribute is composed of the following list elements:

CREP

This static attribute contains the local identifier for the related CR for this AR.

AR Type = MS0

This constraint reflects the attributes for this AR if the value of the attribute AR Type is set to MS0.

Error Action Flag

This static attribute indicates how the reaction of the DP-Master (Class 1) in case of failed data transfers in the operation mode OPERATE shall be. The reaction shall be performed according to the following rules:

If the data transfer to at least one selected (dependent on the attribute Ignore ACIr in the List of CRL Entries) DP-Slave was erroneous and the Error Action Flag is set to TRUE, the operation mode will change from OPERATE to CLEAR.

If the Error Action Flag is set to FALSE, the DP-Master (Class 1) will remain in the operation mode OPERATE despite erroneous data transfers.

Furthermore this attribute indicates how the transition from the operation mode CLEAR to the operation mode OPERATE shall be performed:

If the Error Action Flag set to TRUE the transition shall only be performed if the data transfer to the selected DP-Slaves has been successfully executed.

If the Error Action Flag set to FALSE the transition shall be performed independently of a successfully executed data transfer to all activated DP-Slaves.

This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FALSE

Min Slave Interval

This static attribute indicates the smallest allowed period of time between two consecutive DP-Slave poll cycles. This ensures that the sequence of cyclic service requests from the DP-Master (Class 1) can be handled by the DP-Slave. This period of time will be complied by the DP-Master (Class 1) at the beginning of each cyclic data transfer. This time interval is not guaranteed for the User Global Control services and for acyclic MS1 AR requests. This attribute shall be set accordingly with the Load ARL service.

Allowed values: 1 to $2^{16}-1$

Time Base: 100µs

Data Control Time

This static attribute indicates the period of time in which a Global Control shall be send indicating the operation mode of the DP-Master (Class 1) and at least one successful interaction shall be executed with every activated DP-Slave. This attribute shall be set accordingly with the Load ARL service.

Allowed values: 1 to $2^{16}-1$

Time Base: 10ms

Max User Global Control

This static attribute indicates the maximum number of Global Control requests, which may be submitted by the User before the Global Control confirmations have been received. This attribute describes the ability of the DP-Master (Class 1). A practicable value for Max User Global Control is 16. For each of the 8 Slave groups one Sync and one Freeze Command may be started at the same time. This attribute shall be set accordingly with the Load ARL service.

Allowed values: 1 to 255

Max Diag Data Len

This static attribute indicates the maximum length of Diag Data of all assigned DP-Slaves. This attribute shall be set accordingly with the Load ARL service.

Allowed values: 6 to 244

AR Type = MS1

This constraint reflects the attributes for this AR if the value of the attribute AR Type is set to MS1.

Extra Alarm SAP

This static attribute selects the CR for the Alarm Ack service. If the attribute is set to FALSE the DP-Master (Class 1) acknowledges alarms via the CR for the read write services. If the attribute is set to TRUE the DP-Master (Class 1) acknowledges alarms via a different CR. This attribute shall be set accordingly with the Load ARL service.

Allowed values: TRUE, FLASE

Max Channel Data Length

This static attribute indicates the maximum length of the MS1 AR PDUs (Length parameter and 4 Byte Header) of the corresponding DP-Slave. This attribute shall be set accordingly with the Load ARL service.

Allowed values: 4 to 244

Alarm Mode

This static attribute indicates the maximum number of possible active alarms of the corresponding DP-Slave. This attribute shall be set accordingly with the Load ARL service. Table 237 shows the allowed values for this attribute.

Table 237 – Alarm Mode

Value	Meaning
0	1 alarm of each type
1	2 alarms in total
2	4 alarms in total
3	8 alarms in total
4	12 alarms in total
5	16 alarms in total
6	24 alarms in total
7	32 alarms in total

MS1 Timeout

This attribute contains the value for the Timeout of the acyclic communication on the MS1 AR and specifies the maximum period of time it may take a DP-Slave to provide a response. The time contained in this attribute is the maximum time of the assigned DP-Slaves.

Attribute Type: Unsigned16

Allowed values: 1 to $2^{16}-1$

Time Base: 10ms

AR Type = MM1

This constraint reflects the attributes for this AR if the value of the attribute AR Type is set to MM1.

Poll Timeout

This static attribute indicates the maximum period of time in which the requester shall fetch the response. This attribute shall be set accordingly with the Load ARL service.

Allowed values: 1 to $2^{16}-1$

Time Base: 1ms

Ident Number

This static attribute contains the Ident Number of the DP-Master (Class 1). This attribute shall be set accordingly with the Load ARL service.

Allowed values: 1 to $2^{16}-1$

AR Name

This static attribute contains the name of the related ARL Entry. This attribute shall be set accordingly with the Load ARL service.

9.3.5.1.2.2 Invocation of ARL DP-Master (Class 1) object

For the invocation of the ARL DP-Master (Class 1) object the following rules have to be considered:

- Only one ARL DP-Master (Class 1) object shall be invoked in a DP-Master (Class 1).
- Only one MS0 Entry shall be contained in the List of ARL Entries.
- Only one MM1 Entry shall be contained in the List of ARL Entries.
- Only one MM2 Entry shall be contained in the List of ARL Entries.

9.3.5.1.3 ARL DP-Master (Class 2) Class Specification

The ARL DP-Master (Class 2) object is described by the following template:

DP ASE:	AR ASE
CLASS:	ARL DP-Master (Class 2)
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1. (m) Key Attribute:	Implicit
2. (m) Attribute:	List of ARL Entries
2.1. (m) Attribute:	AREP
2.2. (m) Attribute:	AR Type
2.3. (m) Attribute:	List of related CRL Entries
2.3.1. (m) Attribute:	CREP
2.4. (c) Constraint:	AR Type = MM1
2.4.1 (m) Attribute:	Poll Timeout
2.5 (o) Attribute:	AR Name
SERVICES:	
1. (m) MgtService:	Load ARL
2. (o) MgtService:	Get ARL

9.3.5.1.3.1 Attributes

Implicit

The attribute Implicit indicates that the ARL DP-Master (Class 2) object is implicitly addressed by the service.

List of ARL Entries

This attribute is composed of the following list elements:

AREP

This static attribute contains the local identifier of this AR.

AR Type

This static attribute specifies the type of this AR.

Allowed values: MS0, MS2, MM1, MM2

List of related CRL Entries

This attribute is composed of the following list elements:

CREP

This static attribute contains the local identifier for the related CR for this AR.

AR Type = MM1

This constraint reflects the attributes for this AR if the value of the attribute AR Type is set to MM1.

Poll Timeout

This static attribute indicates the maximum period of time in which the responder shall provide the response. This attribute shall be set accordingly with the Load ARL service.

Allowed values: 1 to $2^{16}-1$

Time Base: 1ms

AR Name

This static attribute contains the name of the related ARL Entry. This attribute shall be set accordingly with the Load ARL service.

9.3.5.1.3.2 Invocation of ARL DP-Master (Class 2) object

For the invocation of the ARL DP-Master (Class 2) object the following rules have to be considered:

- Only one ARL DP-Master (Class 2) object shall be invoked in a DP-Master (Class 2).
- Only one MM1 Entry shall be contained in the List of ARL Entries.
- Only one MM2 Entry shall be contained in the List of ARL Entries.

9.3.5.2 Communication Relationship List Class Specification**9.3.5.2.1 CRL DP-Slave Class Specification**

The CRL DP-Slave object is described by the following template:

DP ASE:	AR ASE
CLASS:	CRL DP-Slave
CLASS ID:	not used
PARENT CLASS:	TOP

ATTRIBUTES:

1. (m) Key Attribute: Implicit
2. (m) Attribute: List of CRL Entries
- 2.1. (m) Attribute: CREP
- 2.2. (m) Attribute: CR Type
- 2.3. (m) Attribute: Related AREP
- 2.4. (m) Attribute: S_SAP_index
- 2.5. (m) Attribute: D_addr
- 2.6. (m) Attribute: Service_activate
- 2.7. (m) Attribute: Indication_mode
- 2.8. (m) Attribute: Max_DLSDU_length_req_low
- 2.9. (m) Attribute: Max_DLSDU_length_req_high
- 2.10. (m) Attribute: Max_DLSDU_length_ind_low
- 2.11. (m) Attribute: Max_DLSDU_length_ind_high
- 2.12. (o) Attribute: CR Name

SERVICES:

1. (m) MgtService: Load CRL
2. (o) MgtService: Get CRL

9.3.5.2.1.1 Attributes

Implicit

The attribute Implicit indicates that the CRL DP-Slave object is implicitly addressed by the service.

List of CRL Entries

This attribute is composed of the following list elements:

CREP

This static attribute contains the local identifier of this CR. This attribute shall be set accordingly with the Load CRL service.

CR Type

This static attribute indicates the type of this CR. This attribute is dependent of AR Type of the related AR. This attribute shall be set accordingly with the Load CRL service. For each AR Type the correspondent CRL entries shall be set. Table 238 shows the allowed values for this attribute.

Related AREP

This static attribute contains the local identifier for the related AR for this CR. This attribute shall be set accordingly with the Load CRL service.

S_SAP_index

This static attribute contains the identifier for the related Service Access Point of the local Data Link Layer. The value for this attribute has to be set correspondent to the attributes CR Type of this CR and to the attribute AR Type of the related AR. This attribute shall be set accordingly with the Load CRL service. Table 238 shows the allowed values for this attribute.

Table 238 – S_SAP_index

AR Type	CR Type	S_SAP_index	Remark
MS 0	BBUB	NIL	mandatory for the MS0 AR
MS 0	BQUU	61, 62	mandatory for the MS0 AR
MS 0	BBUU	60	mandatory for the MS0 AR
MS 0	QQUU-OM	58	mandatory for the MS0 AR
MS 0	QBCB	55, 56, 57, 59, 60	mandatory for the MS0 AR
MS 1	QQCB-CO	50	optional for the MS1 AR
MS 1	QQCB-CO	51	mandatory for the MS1 AR
MS 1	QQUU-CO	60	mandatory for the MS1 AR
MS 2	QQCB-CO	0 to 48	mandatory for each MS2 AR
MS 2	QQUU-CO	49	mandatory for the MS2 AR

D_addr

This dynamic attribute contains the DL address of the remote DP-Master. The value for this attribute has to be set with the Load CRL service correspondent to the attributes CR Type and S_SAP_index of this CR and to the attribute AR Type of the related AR. Table 239 shows the allowed values for this attribute used with the Load CRL service. Furthermore this attribute is influenced by the AR state machine dependent of the attribute CR Type of this CR and of the attribute AR Type of the related AR. Table 239 shows the allowed values for this attribute for the modification of the AR state machine.

Table 239 – D_addr

AR Type	CR Type	S_SAP_index	D_addr as loaded with the Load CRL service	D_addr modified by the AR state machine
MS 0	BBUB	NIL	None	0 to 125
MS 0	BQUU	61, 62	All	-
MS 0	BBUU	60	All	-
MS 0	QQUU-OM	58	None	0 to 125
MS 0	QBCB	55, 59, 60	All	-
MS 0	QBCB	56, 57	None	All
MS 1	QQCB-CO	50, 51	None	0 to 125
MS 1	QQUU-CO	60	All	-
MS 2	QQCB-CO	0 to 48	None	0 to 125
MS 2	QQUU-CO	49	All	-

where

None means that the local Service Access Point shall not be activated

All means that the local Service Access Point shall be activated with no access protection

0 to 125 means that the local Service Access Point shall be activated with the corresponding access protection

- means that the AR state machines shall not modify the attribute D_addr

Service_activate

This static attribute indicates the DL services which shall be activated for the S_SAP_index of this CRL entry.

The value for this attribute has to be set with the Load CRL service correspondent to the attributes CR Type and S_SAP_index of this CR and to the attribute AR Type of the related AR. Table 240 shows the allowed values for this attribute.

Table 240 – Service_activate

AR Type	CR Type	S_SAP_index	Service_activate
MS 0	BBUB	NIL	SRD
MS 0	BQUU	61, 62	SRD
MS 0	BBUU	60	SRD
MS 0	QQUU-OM	58	SDN
MS 0	QBCB	55, 56, 57, 59, 60	SRD
MS 1	QQCB-CO	50	SRD
MS 1	QQCB-CO	51	SRD
MS 1	QQUU-CO	60	SRD
MS 2	QQCB-CO	0 to 48	SRD
MS 2	QQUU-CO	49	SRD

Indication_mode

This static attribute shall only be present if the attribute Service_activate of this CRL entry has the value SRD. It indicates the Indication_mode which the DLL shall use to convey the indications of the SRD service. The value for this attribute has to be set with the Load CRL service correspondent to the attributes CR Type and S_SAP_index of this CR and to the attribute AR Type of the related AR. Table 241 shows the allowed values for this attribute.

Table 241 – Indication_mode

AR Type	CR Type	S_SAP_index	Indication_mode
MS 0	BBUB	NIL	All
MS 0	BQUU	61, 62	All
MS 0	BBUU	60	All
MS 0	QBCB	55, 56, 57, 59	Data
MS 0	QBCB	60	All
MS 1	QQCB-CO	50, 51	Data
MS 1	QQUU-CO	60	All
MS 2	QQCB-CO	0 to 48	Data
MS 2	QQUU-CO	49	Data

Max_DLSDU_length_req_low

This static attribute indicates the maximum length for the DLSDU with low priority as requester. The value for this attribute has to be set with the Load CRL service correspondent to the attributes CR Type and S_SAP_index of this CR and to the attribute AR Type of the related AR. Table 242 shows the allowed values for this attribute.

Table 242 – Max_DLSDU_length_req_low

AR Type	CR Type	S_SAP_index	Max_DLSDU_length_req_low
MS 0	BBUB	NIL	0 to 244
MS 0	BQUU	61, 62	0
MS 0	BBUU	60	6 to 244
MS 0	QBCB	55	0
MS 0	QBCB	56, 57, 59	0 to 244
MS 0	QQUU-OM	58	0
MS 1	QQCB-CO	50	4
MS 1	QQCB-CO	51	4 to 244
MS 1	QQUU-CO	60	6 to 244
MS 2	QQCB-CO	0 to 48	48 to 244
MS 2	QQUU-CO	49	4

Max_DLSDU_length_req_high

This static attribute indicates the maximum length for the DLSDU with high priority as requester. The value for this attribute has to be set with the Load CRL service correspondent to the attributes CR Type and S_SAP_index of this CR and to the attribute AR Type of the related AR. Table 243 shows the allowed values for this attribute.

Table 243 – Max_DLSDU_length_req_high

AR Type	CR Type	S_SAP_index	Max_DLSDU_length_req_high
MS 0	BBUB	NIL	0 to 244
MS 0	BQUU	61, 62	0
MS 0	BBUU	60	0
MS 0	QBCB	55, 56, 57, 59	0
MS 0	QQUU-OM	58	0
MS 1	QQCB-CO	50, 51	0
MS 1	QQUU-CO	60	0
MS 2	QQCB-CO	0 to 48	0
MS 2	QQUU-CO	49	0

Max_DLSDU_length_ind_low

This static attribute indicates the maximum length for the DLSDU with low priority for indication. The value for this attribute has to be set with the Load CRL service correspondent to the attributes CR Type and S_SAP_index of this CR and to the attribute AR Type of the related AR. Table 244 shows the allowed values for this attribute.

Table 244 – Max_DLSDU_length_ind_low

AR Type	CR Type	S_SAP_index	Max_DLSDU_length_ind_low
MS 0	BBUB	NIL	0
MS 0	BQUU	61, 62	0
MS 0	BBUU	60	0
MS 0	QBCB	55	0
MS 0	QBCB	56, 57, 59	0
MS 0	QQUU-OM	58	0
MS 1	QQCB-CO	50	4
MS 1	QQCB-CO	51	4 to 244
MS 1	QQUU-CO	60	0
MS 2	QQCB-CO	0 to 48	4 to 244
MS 2	QQUU-CO	49	48 to 64

Max_DLSDU_length_ind_high

This static attribute indicates the maximum length for the DLSDU with high priority for indication. The value for this attribute has to be set with the Load CRL service correspondent to the attributes CR Type and S_SAP_index of this CR and to the attribute AR Type of the related AR. Table 245 shows the allowed values for this attribute.

Table 245 – Max_DLSDU_length_ind_high

AR Type	CR Type	S_SAP_index	Max_DLSDU_length_ind_high
MS 0	BBUB	NIL	0 to 244
MS 0	BQUU	61	7 to 244
MS 0	BQUU	62	1 to 244
MS 0	BBUU	60	0
MS 0	QBCB	55	4 to 244
MS 0	QBCB	56, 57, 59	0
MS 0	QQUU-OM	58	2
MS 1	QQCB-CO	50, 51	0
MS 1	QQUU-CO	60	0
MS 2	QQCB-CO	0 to 48	0
MS 2	QQUU-CO	49	0

CR Name

This static attribute contains the name of the related CRL Entry. This attribute shall be set accordingly with the Load CRL service.

9.3.5.2.1.2 Invocation of CRL DP-Slave object

For the invocation of the CRL DP-Slave object the following rule has to be considered:

- Only one CRL DP-Slave object shall be invoked in a DP-Slave.

9.3.5.2.2 CRL DP-Master (Class 1) Class Specification

The CRL DP-Master (Class 1) object is described by the following template:

DP ASE:	AR ASE
CLASS:	CRL DP-Master (Class 1)
CLASS ID:	not used
PARENT CLASS:	TOP

ATTRIBUTES:

1.	(m)	Key Attribute:	Implicit
2.	(m)	Attribute:	List of CRL Entries
2.1.	(m)	Attribute:	CREP
2.2.	(m)	Attribute:	CR Type
2.3.	(m)	Attribute:	Related AREP
2.4.	(m)	Attribute:	S_SAP_index
2.5.	(m)	Attribute:	D_SAP_index
2.6.	(m)	Attribute:	D_addr
2.7.	(m)	Attribute:	Service_activate
2.8.	(m)	Attribute:	Role_in_service
2.9.	(m)	Attribute:	Indication_mode
2.10.	(m)	Attribute:	Max_DLSDU_length_req_low
2.11.	(m)	Attribute:	Max_DLSDU_length_req_high
2.12.	(m)	Attribute:	Max_DLSDU_length_ind_low
2.13.	(m)	Attribute:	Max_DLSDU_length_ind_high
2.14.	(c)	Constraint:	CR Type = BQUU & D_SAP_index = 61
2.14.1.	(m)	Attribute:	Active
2.14.2.	(m)	Attribute:	New Prm
2.14.3.	(m)	Attribute:	Prm Data
2.14.3.1.	(m)	Attribute:	Lock
2.14.3.2.	(m)	Attribute:	Sync
2.14.3.3.	(m)	Attribute:	Freeze
2.14.3.4.	(m)	Attribute:	WD On
2.14.3.5.	(m)	Attribute:	WD Fact 1
2.14.3.6.	(m)	Attribute:	WD Fact 2
2.14.3.7.	(m)	Attribute:	Min Tsdr
2.14.3.8.	(m)	Attribute:	Ident Number
2.14.3.9.	(m)	Attribute:	Group Ident
2.14.3.10.	(o)	Attribute:	DPV1 Enable
2.14.3.11.	(o)	Attribute:	Fail Safe
2.14.3.12.	(o)	Attribute:	WD Base
2.14.3.13.	(o)	Attribute:	Enable Pull Plug Alarm
2.14.3.14.	(o)	Attribute:	Enable Process Alarm
2.14.3.15.	(o)	Attribute:	Enable Diagnosis Alarm
2.14.3.16.	(o)	Attribute:	Enable Manufacturer Specific Alarm
2.14.3.17.	(o)	Attribute:	Enable Status Alarm
2.14.3.18.	(o)	Attribute:	Enable Update Alarm

- 2.14.3.19. (o) Attribute: Check Cfg Mode
- 2.14.3.20. (o) Attribute: Alarm Mode
- 2.14.3.21. (o) Attribute: User Prm Data
- 2.14.3.21.1. (o) Attribute: Device Related User Parameter
- 2.14.3.21.1.1. (o) Attribute: Data Description
- 2.14.3.21.2. (o) Attribute: List of Module Related User Parameter
- 2.14.3.21.2.1. (o) Attribute: Data Description
- 2.15. (c) Constraint: CR Type = BQUU & D_SAP_index = 62
- 2.15.1. (m) Attribute: List of Cfg Data
- 2.15.1.1. (s) Attribute: Cfg Identifier
- 2.15.1.1.1. (m) Attribute: Type
- 2.15.1.1.2. (m) Attribute: Format
- 2.15.1.1.3. (m) Attribute: Consistency
- 2.15.1.1.4. (m) Attribute: Length
- 2.15.1.2.(s) Attribute: Special Cfg Identifier
- 2.15.1.2.1. (m) Attribute: Type
- 2.15.1.2.2. (c) Constraint: Type = Input or Input-Output
- 2.15.1.2.2.1. (m) Attribute: Input Length
- 2.15.1.2.2.2. (m) Attribute: Input Format
- 2.15.1.2.2.3. (m) Attribute: Input Consistency
- 2.15.1.2.3. (c) Constraint: Type = Output or Input-Output
- 2.15.1.2.3.1. (m) Attribute: Output Length
- 2.15.1.2.3.2. (m) Attribute: Output Format
- 2.15.1.2.3.3. (m) Attribute: Output Consistency
- 2.15.1.2.4. (o) Attribute: List of Data Description
- 2.15.1.2.4.1. (m) Attribute: Data Description
- 2.15.1.2.5. (o) Attribute: Manufacturer Specific Data
- 2.16. (c) Constraint: CR Type = BBUI & D_SAP_index = 60
- 2.16.1. (m) Attribute: Diag Upd Delay
- 2.17. (c) Constraint: CR Type = BBUB & D_SAP_index = NIL
- 2.17.1. (m) Attribute: Fail Safe
- 2.17.2. (m) Attribute: DPV1 Supported
- 2.17.3. (m) Attribute: NA to Abort
- 2.17.4. (m) Attribute: Ignore ACIr
- 2.17.5. (m) Attribute: Output Len
- 2.17.6. (m) Attribute: Exp Inp Len
- 2.17.7. (m) Attribute: Data Transfer Status
- 2.17.8. (m) Attribute: System Diagnosis Status
- 2.18. (o) Attribute: CR Name

SERVICES:

- 1. (m) MgtService: Load CRL
- 2. (o) MgtService: Get CRL

9.3.5.2.2.1 Attributes

Implicit

The attribute Implicit indicates that the CRL DP-Master (Class 1) object is implicitly addressed by the service.

List of CRL Entries

This attribute is composed of the following list elements:

CREP

This static attribute contains the local identifier of this CR. This attribute shall be set accordingly with the Load CRL service.

CR Type

This static attribute indicates the type of this CR. This attribute is dependent of AR Type of the related AR. This attribute shall be set accordingly with the Load CRL service. For each AR Type the correspondent CRL entries shall be set. Table 246 shows the allowed values for this attribute.

Related AREP

This static attribute contains the local identifier for the related AR for this CR. This attribute shall be set accordingly with the Load CRL service.

S_SAP_index

This static attribute contains the identifier for the related Service Access Point of the local Data Link Layer. The value for this attribute has to be set correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. This attribute shall be set accordingly with the Load CRL service. Table 246 shows the allowed values for this attribute.

Table 246 – S_SAP_index

AR Type	CR Type	S_SAP_index	Remark
MS0	BBUB	NIL	mandatory for the MS0 AR
MS0	BQUU	62	mandatory for the MS0 AR
MS0	BBUU	62	mandatory for the MS0 AR
MS0	QQUU-OM	62	mandatory for the MS0 AR
MS1	QQCB-CO	51	mandatory for the MS1 AR
MS1	QQUU-OO	62	mandatory for the MS1 AR
MM1	QQCB-CL	54	mandatory for the MM1 AR
MM2	QQUU-OM	54	mandatory for the MM2 AR

D_SAP_index

This static attribute contains the identifier for the related Service Access Point of the remote Data Link Layer. The value for this attribute has to be set correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. This attribute shall be set accordingly with the Load CRL service. Table 247 shows the allowed values for this attribute.

Table 247 – D_SAP_index

AR Type	CR Type	D_SAP_index	Remark
MS0	BBUB	NIL	mandatory for the MS0 AR
MS0	BQUU	61, 62	mandatory for the MS0 AR
MS0	BBUU	60	mandatory for the MS0 AR
MS0	QQUU-OM	58	mandatory for the MS0 AR
MS1	QQCB-CO	50	optional for each MS1 AR
MS1	QQCB-CO	51	mandatory for the MS1 AR
MS1	QQUU-OO	60	mandatory for the MS1 AR
MM1	QQCB-CL	54	mandatory for the MM1 AR
MM2	QQUU-OM	54	mandatory for the MM2 AR

D_addr

This dynamic attribute contains the DL address of the remote communication partner. The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. Table 248 shows the allowed values for this attribute used with the Load CRL service. Furthermore this attribute is influenced by the AR state machine dependent of the attribute CR Type of this CR and of the attribute AR Type of the related AR. Table 248 shows the allowed values for this attribute for the modification of the AR state machine.

Table 248 – D_addr

AR Type	CR Type	S_SAP_index	D_addr as loaded with the Load CRL service	D_addr modified by the AR state machine
MS 0	BBUB	NIL	0 to 126	-
MS 0	BQUU	62	0 to 126	-
MS 0	BBUU	62	0 to 126	-
MS 0	QQUU-OM	62	0 to 127	-
MS 1	QQCB-CO	51	0 to 126	-
MS 1	QQUU-CO	62	0 to 126	-
MM 1	QQCB-CL	54	All	0 to 125
MM 2	QQUU-OM	54	All	0 to 125

where

All	means that the local Service Access Point shall be activated with no access protection
0 to 127	means that the attribute D_addr of the local Service Access Point shall be set with the corresponding remote DL address
-	means that the AR state machines shall not modify the attribute D_addr

Service_activate

This static attribute indicates the DL services which shall be activated for the S_SAP_index of this CRL entry.

The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. Table 249 shows the allowed values for this attribute.

Table 249 – Service_activate

AR Type	CR Type	S_SAP_index	Service_activate
MS 0	BBUB	NIL	SRD
MS 0	BQUU	62	SRD
MS 0	BBUU	62	SRD
MS 0	QQUU-OM	62	SDN
MS 1	QQCB-CO	51	SRD
MS 1	QQUU-CO	62	SRD
MM 1	QQCB-CL	54	SRD
MM 2	QQUU-OM	54	SDN

Role_in_service

This static attribute indicates the role which shall be activated for the S_SAP_index of this CRL entry. The following values are specified:

- Initiator: The station initiates the respective service exclusively
- Responder: The station response to service exclusively

The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. Table 250 shows the allowed values for this attribute.

Table 250 – Role_in_service

AR Type	CR Type	S_SAP_index	Role_in_service
MS 0	BBUB	NIL	Initiator
MS 0	BQUU	62	Initiator
MS 0	BBUU	62	Initiator
MS 0	QQUU-OM	62	Initiator
MS 1	QQCB-CO	51	Initiator
MS 1	QQUU-CO	62	Initiator
MM 1	QQCB-CL	54	Responder
MM 2	QQUU-OM	54	Responder

Indication_mode

This conditional and static attribute shall only be present if the attribute Service_activate of this CRL entry has the value SRD and if the attribute Role_in_service of this CRL entry has the value Responder. It indicates the Indication_mode which the DLL shall use to convey the indications of the SRD service. The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. Table 251 shows the allowed values for this attribute.

Table 251 – Indication_mode

AR Type	CR Type	S_SAP_index	Indication_mode
MS 0	BBUB	NIL	-
MS 0	BQUU	62	-
MS 0	BBUU	62	-
MS 0	QQUU-OM	62	-
MS 1	QQCB-CO	51	-
MS 1	QQUU-CO	62	-
MM 1	QQCB-CL	54	Data
MM 2	QQUU-OM	54	-
where			
- means that the attribute Indication_mode is not relevant for this CR Type			

Max_DLSDU_length_req_low

This static attribute indicates the maximum length for the DLSDU with low priority as requester/responder. The value for this attribute has to be set with the Load CRL service correspondent to the attributes CR Type and D_SAP_index of this CR and to the attribute AR Type of the related AR. Table 252 shows the allowed values for this attribute.

Table 252 – Max_DLSDU_length_req_low

AR Type	CR Type	D_SAP_index	Max_DLSDU_length_req_low
MS0	BBUB	NIL	0
MS0	BQUU	61, 62	0
MS0	BBUU	60	0
MS0	QQUU-OM	58	0
MS1	QQCB-CO	50	4
MS1	QQCB-CO	51	4 to 244
MS1	QQUU-CO	60	0
MM1	QQCB-CL	54	4 to 244
MM2	QQUU-OM	54	0

Max_DLSDU_length_req_high

This static attribute indicates the maximum length for the DLSDU with high priority as requester/responder. The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type and D_SAP_index of this CR and to the attribute AR Type of the related AR. Table 253 shows the allowed values for this attribute.

Table 253 – Max_DLSDU_length_req_high

AR Type	CR Type	D_SAP_index	Max_DLSDU_length_req_high
MS0	BBUB	NIL	0 to 244
MS0	BQUU	61	7 to 244
MS0	BQUU	62	1 to 244
MS0	BBUU	60	0
MS0	QQUU-OM	58	2
MS1	QQCB-CO	50, 51	0
MS1	QQUU-OO	60	0
MM1	QQCB-CL	54	0
MM2	QQUU-OM	54	0

Max_DLSDU_length_ind_low

This static attribute indicates the maximum length for the DLSDU with low priority for indication. The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type and D_SAP_index of this CR and to the attribute AR Type of the related AR. Table 254 shows the allowed values for this attribute.

Table 254 – Max_DLSDU_length_ind_low

AR Type	CR Type	D_SAP_index	Max_DLSDU_length_ind_low
MS0	BBUB	NIL	0 to 244
MS0	BQUU	61,62	0
MS0	BBUU	60	6 to 244
MS0	QQUU-OM	58	0
MS1	QQCB-CO	50	4
MS1	QQCB-CO	51	4 to 244
MS1	QQUU-OO	60	6 to 244
MM1	QQCB-CL	54	4 to 244
MM2	QQUU-OM	54	2

Max_DLSDU_length_ind_high

This static attribute indicates the maximum length for the DLSDU with high priority for indication. The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type and D_SAP_index of this CR and to the attribute AR Type of the related AR. Table 255 shows the allowed values for this attribute.

Table 255 – Max_DLSDU_length_ind_high

AR Type	CR Type	D_SAP_index	Max_DLSDU_length_ind_high
MS0	BBUB	NIL	0 to 244
MS0	BQUU	61,62	0
MS0	BBUU	60	0
MS0	QQUU-OM	58	0
MS1	QQCB-CO	50, 51	0
MS1	QQUU-OO	60	0
MM1	QQCB-CL	54	0
MM2	QQUU-OM	54	0

CR Type = BQUU & D_SAP_index = 61

This constraint reflects the attributes for this CR if the value of the attribute CR Type is set to BQUU and the attribute D_SAP_index is set to 61.

Active

This dynamic attribute indicates the state of interactions to the correspondent DP-Slave. The attribute set to FALSE indicates that the DP-Master (Class 1) shall stop the interactions to the correspondent DP-Slave. In this state it is allowed to change the CRL entries of this DP-Slave with the Load CRL service. If this attribute is set to TRUE the DP-Master (Class 1) shall start the interactions to the correspondent DP-Slave. In this state only the attribute Prm Data may be changed with the Load CRL service (see also the attribute New Prm). This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

New Prm

This dynamic attribute indicates to the MS0 AR state machine to send new parameterisation data during the data exchange mode to the correspondent DP-Slave. The following rules shall be applied: If the attribute is set to FALSE the DP-Master (Class 1) transfers the input, output data to the correspondent DP-Slave. If this attribute is set to TRUE and the attribute DPV1 Supported of the related CRL entry has the value FALSE the DP-Master (Class 1) transfers for one DP-Slave poll cycle new parameterisation data to the correspondent DP-Slave. The MS0 AR state machine sets the attribute to FALSE if the transfer was successful.

This attribute shall be set accordingly with the Load CRL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: TRUE, FALSE

Prm Data

This dynamic attribute is an aggregate of the parameterisation data of the correspondent DP-Slave and is composed of the following elements. This attribute shall be set accordingly with the Load CRL service.

Lock

This dynamic attribute is used by the MS0 AR state machine to assign or release the correspondent DP-Slave. This attribute is set to LOCK if the attribute Active of the related CRL entry is set to TRUE. This attribute is set to UNLOCK if the attribute Active of the related CRL entry is set to FALSE, or due to failures during the data exchange mode. This attribute shall be set to LOCK with the Load CRL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: LOCK, UNLOCK

Sync

This dynamic attribute indicates to the correspondent DP-Slave that it shall participate in the synchronisation of the Output Data (Sync mode). The DP-Slave shall check whether it supports the requested functionality and if the DP-Slave does not support this function it shall respond with an error entry in the Slave Diagnosis (Not Supported). This attribute shall be set accordingly with the Load CRL service. Table 256 shows the allowed values for this attribute.

Table 256 – Sync

Value	Meaning
FALSE	Sync ability not requested for this DP-Slave
TRUE	Sync ability requested for this DP-Slave

Freeze

This dynamic attribute indicates to the correspondent DP-Slave that it shall participate in the synchronisation of the Input Data (Freeze mode). The DP-Slave shall check whether it supports the requested functionality and if the DP-Slave does not support this function it shall respond with an error entry in the Slave Diagnosis (Not Supported). This attribute shall be set accordingly with the Load CRL service. Table 257 shows the allowed values for this attribute.

Table 257 – Freeze

Value	Meaning
FALSE	Freeze ability not requested for this DP-Slave
TRUE	Freeze ability requested for this DP-Slave

WD On

This dynamic attribute indicates if the watchdog control shall be activated at the correspondent DP-Slave. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

WD Fact 1, WD Fact 2

These dynamic attributes contain the values for the monitoring of the DP-Master (Class 1) at the MS0 AR (watchdog control (T_{WD})) at the correspondent DP-Slave. The watchdog control in a DP-Slave takes care that, if the DP-Master fails, the outputs

will be set in the safe state after the expiration of this time. The time is calculated according to equation:

$$T_{WD} = \text{Watchdog Time Base} \times \text{WD Fact 1} \times \text{WD Fact 2}$$

If the attribute DPV1 Supported has the value FALSE the Watchdog Time Base shall be set to 10 ms.

If the attribute DPV1 Supported has the value TRUE the Watchdog Time Base shall be set to the value 1ms if the attribute WD Base has the value TRUE. If the attribute DPV1 Supported has the value TRUE the Watchdog Time Base shall be set to the value 10ms if the attribute WD Base has the value FALSE. These attributes shall be set accordingly with the Load CRL service.

Allowed values: 1 to 255

Min TSDR

This dynamic attribute contains the value for the Minimum Station Delay Responder time of the DLL of the correspondent DP-Slave. If this attribute is equal zero, the previous value of the Min TSDR remains unchanged. These attributes shall be set accordingly with the Load CRL service.

Allowed values: 0 to 255

Ident Number

This static attribute contains the Ident Number of the correspondent DP-Slave. The DP-Slave accepts the parameterisation only if the transmitted Ident Number is identical with the attribute Ident Number of the MS0 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service.

Allowed values: 0 to $2^{16}-1$

Group Ident

This dynamic attribute indicates the group(s) to which the correspondent DP-Slave shall be assigned by the DP-Master (Class 1) in the parameterisation. The Group Ident distinguishes 8 groups. If the Group Ident is set to No Group, the DP-Slave is not assigned to any group. The DP-Slave stores the value of this attribute in the attribute Group Identifier of the MS0 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service.

Allowed values: Group 1 to Group 8, No Group and combinations

DPV1 Enabled

This conditional and dynamic attribute indicates if the extended functionality shall be activated at the correspondent DP-Slave. This attribute shall only be present if the attribute DPV1 Supported in the related MS0 AR entry in the List of ARL Entries of the DP-Slave has the value TRUE. This attribute shall be set accordingly with the Load CRL service. Table 258 shows the allowed values for this attribute.

Table 258 – DPV1 Enabled

Value	Meaning
FALSE	The extended functionality related with the attribute DPV1 Supported is disabled.

TRUE	The extended functionality related with the attribute DPV1 Supported is enabled.

Fail Safe

This conditional and dynamic attribute indicates to the correspondent DP-Slave if the DP-Master (Class 1) supports the Fail Safe mode. If the DP-Slave requires the Fail Safe mode and the DP-Master (Class 1) does not support this mode the DP-Slave shall respond with an error entry in the Slave Diagnosis (Prm Fault). The DP-Slave stores the value of this attribute in the attribute Fail Safe of the MS0 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service. Table 259 shows the allowed values for this attribute.

Table 259 – Fail Safe

Value	Meaning
FALSE	The DP-Master (Class 1) does not support the Fail Safe mode.
TRUE	The DP-Master (Class 1) supports the Fail Safe mode.

WD Base

This conditional and dynamic attribute contains the Time Base value for the watchdog control according to the parameterisation for the correspondent DP-Slave. The DP-Slave stores the value of this attribute in the attribute WD Base of the MS0 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service. Table 260 shows the allowed values for this attribute.

Table 260 – WD Base

Value	Meaning
FALSE	The Watchdog Time Base is 10ms.
TRUE	The Watchdog Time Base is 1ms.

Enable Pull Plug Alarm

This conditional and dynamic attribute indicates if the transmission of the Pull Plug Alarm is enabled at the correspondent DP-Slave. The DP-Slave stores the value of this attribute in the attribute Pull Plug Alarm enabled of the MS1 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

Enable Process Alarm

This conditional and dynamic attribute indicates if the transmission of the Process Alarm is enabled at the correspondent DP-Slave. The DP-Slave stores the value of this attribute in the attribute Process Alarm enabled of the MS1 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

Enable Diagnosis Alarm

This conditional and dynamic attribute indicates if the transmission of the Diagnosis Alarm is enabled at the correspondent DP-Slave. The DP-Slave stores the value of this attribute in the attribute Diagnosis Alarm enabled of the MS1 AR entry in the List of

ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

Enable Manufacturer Specific Alarm

This conditional and dynamic attribute indicates if the transmission of the Manufacturer Specific Alarm is enabled at the correspondent DP-Slave. The DP-Slave stores the value of this attribute in the attribute Manufacturer Specific Alarm enabled of the MS1 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

Enable Status Alarm

This conditional and dynamic attribute indicates if the transmission of the Status Alarm is enabled at the correspondent DP-Slave. The DP-Slave stores the value of this attribute in the attribute Status Alarm enabled of the MS1 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

Enable Update Alarm

This conditional and dynamic attribute indicates if the transmission of the Update Alarm is enabled at the correspondent DP-Slave. The DP-Slave stores the value of this attribute in the attribute Update Alarm enabled of the MS1 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

Check Cfg Mode

This conditional and dynamic attribute indicates how to check of the Cfg Data at the correspondent DP-Slave. The value of this attribute is passed to the application of the DP-Slave with the Check Cfg service and influences the check of the Cfg Data (see 9.2.5.3.3). This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

Alarm Mode

This conditional and dynamic attribute indicates the maximum number of possible active alarms between the DP-Master (Class 1) and this DP-Slave. The DP-Slave stores the value of this attribute in the attribute Alarm Mode enabled of the MS1 AR entry in the List of ARL Entries of the DP-Slave. This attribute shall be set accordingly with the Load CRL service. Table 261 shows the allowed values for this attribute.

Table 261 – Alarm Mode

Value	Meaning
0	1 alarm of each type
1	2 alarms in total
2	4 alarms in total
3	8 alarms in total
4	12 alarms in total

5	16 alarms in total
6	24 alarms in total
7	32 alarms in total

User Prm Data

This optional and dynamic attribute is an aggregate of the Device and/or List of Module Related User Parameter of the correspondent DP-Slave and is composed of the following elements. This attribute shall be set accordingly with the Load CRL service.

Device Related User Parameter

This attribute is composed of following elements:

Data Description

This optional and dynamic attribute contains either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in 5.2.1 is used. This attribute shall be set accordingly with the Load CRL service.

List of Module Related User Parameter

This attribute is composed of following list elements (each element corresponds to a module).

Data Description

This optional and dynamic attribute contains either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in 5.2.1 is used. This attribute shall be set accordingly with the Load CRL service.

CR Type = BQUU & D_SAP_index = 62

This constraint reflects the attributes for this CR if the value of the attribute CR Type is set to BQUU and the attribute D_SAP_index is set to 62.

List of Cfg Data

This dynamic attribute is composed of the following list elements:

Cfg Identifier

This static attribute describes the basic configuration format and is composed of the following elements. This attribute shall be set accordingly with the Load CRL service.

Type

This static attribute indicates the type of the related Input/Output Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: Input, Output, Input-Output

Format

This static attribute indicates the format of the related Input/Output Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: Byte, Word

Consistency

This static attribute indicates the consistency of the related Input/Output Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: Byte, Word, whole length

Length

This static attribute indicates the length of the related Input/Output Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: 1 to 32

Special Cfg Identifier

This static attribute describes the special configuration format and is composed of the following elements. This attribute shall be set accordingly with the Load CRL service.

Type

This static attribute indicates the type of the related Input/Output Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: Input, Output, Input-Output

Input Length

This static attribute indicates the length of the related Input Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: 1 to 128

Input Format

This static attribute indicates the format of the related Input Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: Byte, Word

Input Consistency

This static attribute indicates the consistency of the related Input Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: Byte, Word, whole length

Output Length

This static attribute indicates the length of the related Output Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: 1 to 128

Output Format

This static attribute indicates the format of the related Output Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: Byte, Word

Output Consistency

This static attribute indicates the consistency of the related Output Data Element. This attribute shall be set accordingly with the Load CRL service.

Allowed values: Byte, Word, whole length

List of Data Description

This static attribute is composed of the following list elements. This attribute shall be set accordingly with the Load CRL service.

Data Description

This static attribute contains one of the allowed Data Types of the data description of one element of the Extended Input/Output Data object. The allowed Data Types are given in 5.2.1. This attribute shall be set accordingly with the Load CRL service.

Manufacturer Specific Data

This static attribute contains the manufacturer specific data for one Special Cfg Identifier. This attribute shall be set accordingly with the Load CRL service.

CR Type = BBUU & D_SAP_index = 60

This constraint reflects the attributes for this CR if the value of the attribute CR Type is set to BBUU and the attribute D_SAP_index is set to 60.

Diag Upd Delay

The static attribute indicates the number of Diag service requests performed in the state DIAG2 of the DP-Master (Class 1) while Prm Req is set in Diag Data of the correspondent DP-Slave. This allows DP-Slaves with reduced performance to update the Diag Data during the start up phase of the DP system. This attribute shall be set accordingly with the Load CRL service.

Allowed values: 0 to 15 (extendible up to 255)

CR Type = BBUB & D_SAP_index = NIL

This constraint reflects the attributes for this CR if the value of the attribute CR Type is set to BBUB and the attribute D_SAP_index is set to NIL.

Fail Safe

This static attribute indicates if the correspondent DP-Slave supports the Fail Safe functionality. If the attribute is set to FALSE the DP-Master (Class 1) shall send the value 0 for the Output Data object for this DP-Slave in the operation mode CLEAR and OPERATE. If the attribute is set to TRUE the following rules shall apply for the DP-Master (Class 1) and the DP-Slave:

The DP-Master (Class 1) shall convey no values (means Output Data length = 0) for the Output Data object for this DP-Slave in the operation mode CLEAR and OPERATE.

The DP-Slave shall trigger the application to set the values of the Output Data object in a safe state after receiving no values for the Output Data object during the data exchange mode.

This attribute shall be set accordingly with the Load CRL service. Table 262 shows the allowed values for this attribute.

Table 262 – Fail Safe

Value	Meaning
FALSE	Fail Safe functionality not supported by the DP-Slave
TRUE	Fail Safe functionality supported by the DP-Slave

DPV1 Supported

This static attribute indicates if the correspondent DP-Slave supports the extended functionality. If the attribute is set to FALSE the DP-Slave supports no extended functionality. If the attribute is set to TRUE the DP-Slave supports the extended functionality. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

NA to Abort

This static attribute indicates the reaction of the DP-Master (Class 1) if the correspondent DP-Slave does not respond at the MS0 AR in the DIAG2 state and the data exchange mode. If this attribute is set to FALSE the DP-Master (Class 1) continues with the interactions even the DP-Slave does not respond (Status = NA) to a request. If the attribute is set to TRUE the DP-Master (Class 1) releases and re-establish the CRs of the MS0 AR to the correspondent DP-Slave if it does not respond (Status = NA) to a request. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

Ignore ACI

This static attribute in combination with the attribute Error Action Flag in the MS0 AR entry of the List of ARL Entries of the DP-Master (Class 1) allows a DP-Slave specific reaction of the DP-Master (Class 1) in case of failed data transfers. If the attribute is set to FALSE the DP-Master (Class 1) processes the auto-clear function according to the value of the attribute Error Action Flag. If the attribute is set to TRUE the DP-Master (Class 1) ignores the auto-clear function according to the value of the attribute Error Action Flag. This attribute shall be set accordingly with the Load CRL service.

Allowed values: TRUE, FALSE

Output Len

This static attribute indicates the length of the data of the Output Data object of the correspondent DP-Slave. The value for this attribute shall be derived from the attribute List of Cfg Data of the related CRL entry. This attribute shall be set accordingly with the Load CRL service.

Allowed values: 0 to 244

Exp Inp Len

This static attribute indicates the length of the data of the Input Data object of the correspondent DP-Slave expected by the DP-Master (Class 1). If the length of the Input Data of the Input Data object do not correspond with the value contained in this attribute the DP-Master (Class 1) releases and re-establish the CRs of the MS0 AR to the correspondent DP-Slave. The value for this attribute shall be derived from the attribute List of Cfg Data of the related CRL entry. This attribute shall be set accordingly with the Load CRL service.

Allowed values: 0 to 244

Data Transfer Status

This dynamic attribute indicates the status of the data exchange mode of the correspondent DP-Slave. This attribute shall be set to TRUE by the MS0 AR state machine

of the DP-Master (Class 1) if the data exchange with the assigned DP-Slave was successfully or if between two data exchange requests is only one diagnosis request.

In all other cases the attribute shall be set to FALSE. This attribute shall be set to FALSE with the Load CRL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: TRUE, FALSE

System Diagnosis Status

This dynamic attribute indicates the state of the correspondent DP-Slave assigned to the DP-Master (Class 1). This attribute shall be set to TRUE by the MS0 AR state machine of the DP-Master (Class 1) if the Ext Diag Flag of the Diagnosis information of the correspondent the DP-Slave is set or if the correspondent the DP-Slave does not respond or if the correspondent the DP-Slave responds incorrectly so that the MS0 AR has to be re-established.

This attribute shall be set to FALSE by the MS0 AR state machine of the DP-Master (Class 1) after the MS0 AR is successfully re-established or if the correspondent DP-Slave is deactivated by the AP of the DP-Master (Class 1) or the operation mode of the DP-Master (Class 1) is STOP.

This attribute shall be set to FALSE with the Load CRL service. This attribute is further influenced by the MS0 AR state machine.

Allowed values: TRUE, FALSE

CR Name

This static attribute contains the name of the related CRL Entry. This attribute shall be set accordingly with the Load CRL service.

9.3.5.2.2 Invocation of CRL DP-Master (Class 1) object

For the invocation of the CRL DP-Master (Class 1) object the following rule has to be considered:

- Only one CRL DP-Master (Class 1) object shall be invoked in a DP-Master (Class 1).

9.3.5.2.3 CRL DP-Master (Class 2) Class Specification

The CRL DP-Master (Class 2) object is described by the following template:

DP ASE:	AR ASE
CLASS:	CRL DP-Master (Class 2)
CLASS ID:	not used
PARENT CLASS:	TOP

ATTRIBUTES:

1. (m) Key Attribute: Implicit
2. (m) Attribute: List of CRL Entries
 - 2.1. (m) Attribute: CREP
 - 2.2. (m) Attribute: CR Type
 - 2.3. (m) Attribute: Related AREP
 - 2.4. (m) Attribute: S_SAP_index
 - 2.5. (m) Attribute: D_SAP_index
 - 2.6. (m) Attribute: D_addr
 - 2.8. (m) Attribute: Service_activate
 - 2.9. (m) Attribute: Role_in_service
 - 2.10. (m) Attribute: Max_DLSDU_length_req_low
 - 2.11. (m) Attribute: Max_DLSDU_length_req_high
 - 2.12. (m) Attribute: Max_DLSDU_length_ind_low
 - 2.13. (m) Attribute: Max_DLSDU_length_ind_high
 - 2.14. (o) Attribute: CR Name

SERVICES:

1. (m) MgtService: Load CRL
2. (o) MgtService: Get CRL

9.3.5.2.3.1 Attributes

Implicit

The attribute Implicit indicates that the CRL DP-Master (Class 2) object is implicitly addressed by the service.

List of CRL Entries

This attribute is composed of the following list elements:

CREP

This static attribute contains the local identifier of this CR. This attribute shall be set accordingly with the Load CRL service.

CR Type

This static attribute indicates the type of this CR. This attribute is dependent of AR Type of the related AR. This attribute shall be set accordingly with the Load CRL service. For each AR Type the correspondent CRL entries shall be set. Table 263 shows the allowed values for this attribute.

Related AREP

This static attribute contains the local identifier for the related AR for this CR. This attribute shall be set accordingly with the Load CRL service.

S_SAP_index

This static attribute contains the identifier for the related Service Access Point of the local Data Link Layer. The value for this attribute has to be set correspondent to the attribute CR

Type of this CR and to the attribute AR Type of the related AR. This attribute shall be set accordingly with the Load CRL service. Table 263 shows the allowed values for this attribute.

Table 263 – S_SAP_index

AR Type	CR Type	S_SAP_index	Remark
MS0	QBCB	62	mandatory for the MS0 AR
MS2	QQCB-CO	50	mandatory for the MS2 AR
MS2	QQUU-OO	50	mandatory for the MS2 AR
MM1	QQCB-CL	54	mandatory for the MM1 AR
MM2	QQUU-OM	54	mandatory for the MM2 AR

D_SAP_index

This static attribute contains the identifier for the related Service Access Point of the remote Data Link Layer. The value for this attribute has to be set correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. This attribute shall be set accordingly with the Load CRL service.

Table 264 shows the allowed values for this attribute.

Table 264 – D_SAP_index

AR Type	CR Type	D_SAP_index	Remark
MS0	QBCB	55, 56, 57, 59, 60	mandatory for the MS0 AR
MS2	QQCB-CO	0 to 48	mandatory for the MS2 AR
MS2	QQUU-OO	49	mandatory for the MS2 AR
MM1	QQCB-CL	54	mandatory for the MM1 AR
MM2	QQUU-OM	54	mandatory for the MM2 AR

D_addr

This dynamic attribute contains the DL address of the remote communication partner. The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. Table 265 shows the allowed values for this attribute used with the Load CRL service.

Table 265 – D_addr

AR Type	CR Type	S_SAP_in dex	D_addr as loaded with the Load CRL service
MS0	QBCB	62	0 to 126
MS2	QQCB-CO	50	0 to 126
MS2	QQUU-OO	50	0 to 126
MM1	QQCB-CL	54	0 to 125
MM2	QQUU-OM	54	0 to 125, 127
where			
0 to 125, 127 means that the attribute D_addr of the local Service Access Point shall be set with the corresponding remote DL address			

Service_activate

This static attribute indicates the DL services which shall be activated for the S_SAP_index of this CRL entry.

The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. Table 266 shows the allowed values for this attribute.

Table 266 – Service_activate

AR Type	CR Type	S_SAP_index	Service_activate
MS0	QBCB	62	SRD
MS2	QQCB-CO	50	SRD
MS2	QQUU-OO	50	SRD
MM1	QQCB-CL	54	SRD
MM2	QQUU-OM	54	SDN

Role_in_service

This static attribute indicates the role which shall be activated for the S_SAP_index of this CRL entry. The following value is specified:

Initiator: The station initiates the respective service exclusively

The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type of this CR and to the attribute AR Type of the related AR. Table 267 shows the allowed values for this attribute.

Table 267 – Role_in_service

AR Type	CR Type	S_SAP_index	Role_in_service
MS0	QBCB	62	Initiator
MS2	QQCB-CO	50	Initiator
MS2	QQUU-OO	50	Initiator
MM1	QQCB-CL	54	Initiator
MM2	QQUU-OM	54	Initiator

Max_DLSDU_length_req_low

This static attribute indicates the maximum length for the DLSDU with low priority as requester. The value for this attribute has to be set with the Load CRL service correspondent to the attributes CR Type and D_SAP_index of this CR and to the attribute AR Type of the related AR. Table 268 shows the allowed values for this attribute.

Table 268 – Max_DLSDU_length_req_low

AR Type	CR Type	D_SAP_index	Max_DLSDU_length_req_low
MS0	QBCB	55, 56, 57, 59, 60	0
MS2	QQCB-CO	0 to 48	4 to 244
MS2	QQUU-OO	49	48 to 64
MM1	QQCB-CL	54	4 to 244
MM2	QQUU-OM	54	2

Max_DLSDU_length_req_high

This static attribute indicates the maximum length for the DLSDU with high priority as requester. The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type and D_SAP_index of this CR and to the attribute AR Type of the related AR. Table 269 shows the allowed values for this attribute.

Table 269 – Max_DLSDU_length_req_high

AR Type	CR Type	D_SAP_index	Max_DLSDU_length_req_high
MS0	QBCB	55	4 to 244
MS0	QBCB	56, 57, 59, 60	0
MS2	QQCB-CO	0 to 48	0
MS2	QQUU-OO	49	0
MM1	QQCB-CL	54	0
MM2	QQUU-OM	54	0

Max_DLSDU_length_ind_low

This static attribute indicates the maximum length for the DLSDU with low priority for indication. The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type and D_SAP_index of this CR and to the attribute AR Type of the related AR. Table 270 shows the allowed values for this attribute.

Table 270 – Max_DLSDU_length_ind_low

AR Type	CR Type	D_SAP_index	Max_DLSDU_length_ind_low
MS0	QBCB	55	0
MS0	QBCB	56, 57	0 to 244
MS0	QBCB	59	1 to 244
MS0	QBCB	60	6 to 244
MS2	QVCB-CO	0 to 48	48 to 244
MS2	QQUU-OO	49	4
MM1	QVCB-CL	54	4 to 244
MM2	QQUU-OM	54	0

Max_DLSDU_length_ind_high

This static attribute indicates the maximum length for the DLSDU with high priority for indication. The value for this attribute has to be set with the Load CRL service correspondent to the attribute CR Type and D_SAP_index of this CR and to the attribute AR Type of the related AR. Table 271 shows the allowed values for this attribute.

Table 271 – Max_DLSDU_length_ind_high

AR Type	CR Type	D_SAP_index	Max_DLSDU_length_ind_high
MS0	QBCB	55, 56, 57, 59, 60	0
MS2	QVCB-CO	0 to 48	0
MS2	QQUU-OO	49	0
MM1	QVCB-CL	54	0
MM2	QQUU-OM	54	0

CR Name

This static attribute contains the name of the related CRL Entry. This attribute shall be set accordingly with the Load CRL service.

9.3.5.2.3.2 Invocation of CRL DP-Master (Class 2) object

For the invocation of the CRL DP-Master (Class 2) object the following rule has to be considered:

- Only one CRL DP-Master (Class 2) object shall be invoked in a DP-Master (Class 2).

9.3.5.3 AR Service Specification

9.3.5.3.1 Load ARL DP Slave

The service given in Table 272 is used to load the ARL of a DP-Slave.

Table 272 – Load ARL DP Slave

Parameter name	.req	.cnf
Argument	M	
Min Send Timeout	C	
List of Resource Manager Entries	C	
API	C	
SCL	C	
Max Len Data Unit	C	
Send Timeout	C	
AREP	C	
S_SAP_index	C	
In Use	C	
List of S_SAP_indices	C	
S_SAP_index	C	
List of ARL Entries	M	
AREP	M	
AR Type	M	
List of related CRL Entries	M	
CREP	M	
Prm Req	C	
Prm Fault	C	
Cfg Fault	C	
Not Supported	C	
Station Not Ready	C	
Stat Diag	C	
Master Add	C	
Sync Supported	C	
Sync Mode	C	
Freeze Supported	C	
Freeze Mode	C	
WD Enabled	C	
WD Fact 1	C	
WD Fact 2	C	
Min TSDR	C	
Ident Number	C	
Group Identifier	C	
DPV1 Supported	C	

(continued on next page)

DPV1 Enabled	C	
Fail Safe Required	C	
Fail Safe	C	
WD Base	C	
List of Real Cfg Data	M	
Cfg Identifier	S	
Type	M	
Format	M	
Consistency	M	
Length	M	
Special Cfg Identifier	S	
Type	M	
Input Length	C	
Input Format	C	
Input Consistency	C	
Output Length	C	
Output Format	C	
Output Consistency	C	
List of Data Description	O	
Data Description	M	
Manufacturer Specific Data	O	
No Add Change	O	
Pull Plug Alarm supported	C	
Pull Plug Alarm enabled	C	
Process Alarm supported	C	
Process Alarm enabled	C	
Diagnosis Alarm supported	C	
Diagnosis Alarm enabled	C	
Manufacturer Specific Alarm supported	C	
Manufacturer Specific Alarm enabled	C	
Status Alarm supported	C	
Status Alarm enabled	C	
Update Alarm supported	C	
Update Alarm enabled	C	
Alarm Mode supported	C	
Alarm Mode enabled	C	
AR Name	O	
Result(+)		S
Result(-)		S
Status		M

Argument

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

Min Send Timeout

This parameter contains the value for the equivalent attribute of the ARL entry.

List of Resource Manager Entries

This parameter is composed of the following list elements:

API

This parameter contains the value for the equivalent attribute of the ARL entry.

SCL

This parameter contains the value for the equivalent attribute of the ARL entry.

Max Len Data Unit

This parameter contains the value for the equivalent attribute of the ARL entry.

Send Timeout

This parameter contains the value for the equivalent attribute of the ARL entry.

AREP

This parameter contains the value for the equivalent attribute of the ARL entry.

S_SAP_index

This parameter contains the value for the equivalent attribute of the ARL entry.

In Use

This parameter contains the value for the equivalent attribute of the ARL entry.

List of S_SAP_indices

This parameter is composed of the following list elements:

S_SAP_index

This parameter contains the value for the equivalent attribute of the ARL entry.

List of ARL Entries

This parameter is composed of the following list elements:

AREP

This parameter contains the value for the equivalent attribute of the ARL entry.

AR Type

This parameter contains the value for the equivalent attribute of the ARL entry.

List of related CRL Entries

This attribute is composed of the following list elements:

CREP

This parameter contains the value for the equivalent attribute of the ARL entry.

Prm Req

This parameter contains the value for the equivalent attribute of the ARL entry.

Prm Fault

This parameter contains the value for the equivalent attribute of the ARL entry.

Cfg Fault

This parameter contains the value for the equivalent attribute of the ARL entry.

Not Supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Station Not Ready

This parameter contains the value for the equivalent attribute of the ARL entry.

Stat Diag

This parameter contains the value for the equivalent attribute of the ARL entry.

Master Add

This parameter contains the value for the equivalent attribute of the ARL entry.

Sync Supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Sync Mode

This parameter contains the value for the equivalent attribute of the ARL entry.

Freeze Supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Freeze Mode

This parameter contains the value for the equivalent attribute of the ARL entry.

WD Enabled

This parameter contains the value for the equivalent attribute of the ARL entry.

WD Fact 1, WD Fact 2

This parameter contains the value for the equivalent attribute of the ARL entry.

Min TSDR

This parameter contains the value for the equivalent attribute of the ARL entry.

Ident Number

This parameter contains the value for the equivalent attribute of the ARL entry.

Group Identifier

This parameter contains the value for the equivalent attribute of the ARL entry.

DPV1 Supported

This parameter contains the value for the equivalent attribute of the ARL entry.

DPV1 Enabled

This parameter contains the value for the equivalent attribute of the ARL entry.

Fail Safe Required

This parameter contains the value for the equivalent attribute of the ARL entry.

Fail Safe

This parameter contains the value for the equivalent attribute of the ARL entry.

WD Base

This parameter contains the value for the equivalent attribute of the ARL entry.

List of Real Cfg Data

This attribute is composed of the following list elements:

Cfg Identifier

This selection type parameter indicates the type of Cfg Identifier format.

Type

This parameter contains the value for the equivalent attribute of the ARL entry.

Format

This parameter contains the value for the equivalent attribute of the ARL entry.

Consistency

This parameter contains the value for the equivalent attribute of the ARL entry.

Length

This parameter contains the value for the equivalent attribute of the ARL entry.

Special Cfg Identifier

This selection type parameter indicates the type of Cfg Identifier format.

Type

This parameter contains the value for the equivalent attribute of the ARL entry.

Input Length

This parameter contains the value for the equivalent attribute of the ARL entry.

Input Format

This parameter contains the value for the equivalent attribute of the ARL entry.

Input Consistency

This parameter contains the value for the equivalent attribute of the ARL entry.

Output Length

This parameter contains the value for the equivalent attribute of the ARL entry.

Output Format

This parameter contains the value for the equivalent attribute of the ARL entry.

Output Consistency

This parameter contains the value for the equivalent attribute of the ARL entry.

List of Data Description

This attribute is composed of the following list elements:

Data Description

This parameter contains the value for the equivalent attribute of the ARL entry.

Manufacturer Specific Data

This parameter contains the value for the equivalent attribute of the ARL entry.

No Add Change

This parameter contains the value for the equivalent attribute of the ARL entry.

Pull Plug Alarm supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Pull Plug Alarm enabled

This parameter contains the value for the equivalent attribute of the ARL entry.

Process Alarm supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Process Alarm enabled

This parameter contains the value for the equivalent attribute of the ARL entry.

Diagnosis Alarm supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Diagnosis Alarm enabled

This parameter contains the value for the equivalent attribute of the ARL entry.

Manufacturer Specific Alarm supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Manufacturer Specific Alarm enabled

This parameter contains the value for the equivalent attribute of the ARL entry.

Status Alarm supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Status Alarm enabled

This parameter contains the value for the equivalent attribute of the ARL entry.

Update Alarm supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Update Alarm enabled

This parameter contains the value for the equivalent attribute of the ARL entry.

Alarm Mode supported

This parameter contains the value for the equivalent attribute of the ARL entry.

Alarm Mode enabled

This parameter contains the value for the equivalent attribute of the ARL entry.

AR Name

This parameter contains the value for the equivalent attribute of the ARL entry.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.2 Get ARL DP Slave

The service given in Table 273 is used to get the ARL of a DP-Slave.

Table 273 – Get ARL DP Slave

Parameter name	.req	.cnf
Argument	M	
Result(+)		S
Min Send Timeout		C
List of Resource Manager Entries		C
API		C
SCL		C
Max Len Data Unit		C
Send Timeout		C
AREP		C
S_SAP_index		C
In Use		C
List of S_SAP_indices		C
S_SAP_index		C
List of ARL Entries		M
AREP		M
AR Type		M
List of related CRL Entries		M
CREP		M
Prm Req		C
Prm Fault		C
Cfg Fault		C
Not Supported		C
Station Not Ready		C
Stat Diag		C
Master Add		C
Sync Supported		C
Sync Mode		C
Freeze Supported		C
Freeze Mode		C
WD Enabled		C
WD Fact 1		C
WD Fact 2		C
Min TSDR		C
Ident Number		C
Group Identifier		C

(continued on next page)

DPV1 Supported	C
DPV1 Enabled	C
Fail Safe Required	C
Fail Safe	C
WD Base	C
List of Real Cfg Data	M
Cfg Identifier	S
Type	M
Format	M
Consistency	M
Length	M
Special Cfg Identifier	S
Type	M
Input Length	C
Input Format	C
Input Consistency	C
Output Length	C
Output Format	C
Output Consistency	C
List of Data Description	O
Data Description	M
Manufacturer Specific Data	O
No Add Change	O
Pull Plug Alarm supported	C
Pull Plug Alarm enabled	C
Process Alarm supported	C
Process Alarm enabled	C
Diagnosis Alarm supported	C
Diagnosis Alarm enabled	C
Manufacturer Specific Alarm supported	C
Manufacturer Specific Alarm enabled	C
Status Alarm supported	C
Status Alarm enabled	C
Update Alarm supported	C
Update Alarm enabled	C
Alarm Mode supported	C
Alarm Mode enabled	C
AR Name	O
Result(-)	S
Status	M

Argument

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Min Send Timeout

This parameter contains the value of the equivalent attribute of the ARL entry.

List of Resource Manager Entries

This parameter is composed of the following list elements:

API

This parameter contains the value of the equivalent attribute of the ARL entry.

SCL

This parameter contains the value of the equivalent attribute of the ARL entry.

Max Len Data Unit

This parameter contains the value of the equivalent attribute of the ARL entry.

Send Timeout

This parameter contains the value of the equivalent attribute of the ARL entry.

AREP

This parameter contains the value of the equivalent attribute of the ARL entry.

S_SAP_index

This parameter contains the value of the equivalent attribute of the ARL entry.

In Use

This parameter contains the value of the equivalent attribute of the ARL entry.

List of S_SAP_indices

This parameter is composed of the following list elements:

S_SAP_index

This parameter contains the value of the equivalent attribute of the ARL entry.

List of ARL Entries

This parameter is composed of the following list elements:

AREP

This parameter contains the value of the equivalent attribute of the ARL entry.

AR Type

This parameter contains the value of the equivalent attribute of the ARL entry.

List of related CRL Entries

This attribute is composed of the following list elements:

CREP

This parameter contains the value of the equivalent attribute of the ARL entry.

Prm Req

This parameter contains the value of the equivalent attribute of the ARL entry.

Prm Fault

This parameter contains the value of the equivalent attribute of the ARL entry.

Cfg Fault

This parameter contains the value of the equivalent attribute of the ARL entry.

Not Supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Station Not Ready

This parameter contains the value of the equivalent attribute of the ARL entry.

Stat Diag

This parameter contains the value of the equivalent attribute of the ARL entry.

Master Add

This parameter contains the value of the equivalent attribute of the ARL entry.

Sync Supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Sync Mode

This parameter contains the value of the equivalent attribute of the ARL entry.

Freeze Supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Freeze Mode

This parameter contains the value of the equivalent attribute of the ARL entry.

WD Enabled

This parameter contains the value of the equivalent attribute of the ARL entry.

WD Fact 1, WD Fact 2

This parameter contains the value of the equivalent attribute of the ARL entry.

Min TSDR

This parameter contains the value of the equivalent attribute of the ARL entry.

Ident Number

This parameter contains the value of the equivalent attribute of the ARL entry.

Group Identifier

This parameter contains the value of the equivalent attribute of the ARL entry.

DPV1 Supported

This parameter contains the value of the equivalent attribute of the ARL entry.

DPV1 Enabled

This parameter contains the value of the equivalent attribute of the ARL entry.

Fail Safe Required

This parameter contains the value of the equivalent attribute of the ARL entry.

Fail Safe

This parameter contains the value of the equivalent attribute of the ARL entry.

WD Base

This parameter contains the value of the equivalent attribute of the ARL entry.

List of Real Cfg Data

This attribute is composed of the following list elements:

Cfg Identifier

This selection type parameter indicates the type of Cfg Identifier format.

Type

This parameter contains the value of the equivalent attribute of the ARL entry.

Format

This parameter contains the value of the equivalent attribute of the ARL entry.

Consistency

This parameter contains the value of the equivalent attribute of the ARL entry.

Length

This parameter contains the value of the equivalent attribute of the ARL entry.

Special Cfg Identifier

This selection type parameter indicates the type of Cfg Identifier format.

Type

This parameter contains the value of the equivalent attribute of the ARL entry.

Input Length

This parameter contains the value of the equivalent attribute of the ARL entry.

Input Format

This parameter contains the value of the equivalent attribute of the ARL entry.

Input Consistency

This parameter contains the value of the equivalent attribute of the ARL entry.

Output Length

This parameter contains the value of the equivalent attribute of the ARL entry.

Output Format

This parameter contains the value of the equivalent attribute of the ARL entry.

Output Consistency

This parameter contains the value of the equivalent attribute of the ARL entry.

List of Data Description

This attribute is composed of the following list elements.

Data Description

This parameter contains the value of the equivalent attribute of the ARL entry.

Manufacturer Specific Data

This parameter contains the value of the equivalent attribute of the ARL entry.

No Add Change

This parameter contains the value of the equivalent attribute of the ARL entry.

Pull Plug Alarm supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Pull Plug Alarm enabled

This parameter contains the value of the equivalent attribute of the ARL entry.

Process Alarm supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Process Alarm enabled

This parameter contains the value of the equivalent attribute of the ARL entry.

Diagnosis Alarm supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Diagnosis Alarm enabled

This parameter contains the value of the equivalent attribute of the ARL entry.

Manufacturer Specific Alarm supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Manufacturer Specific Alarm enabled

This parameter contains the value of the equivalent attribute of the ARL entry.

Status Alarm supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Status Alarm enabled

This parameter contains the value of the equivalent attribute of the ARL entry.

Update Alarm supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Update Alarm enabled

This parameter contains the value of the equivalent attribute of the ARL entry.

Alarm Mode supported

This parameter contains the value of the equivalent attribute of the ARL entry.

Alarm Mode enabled

This parameter contains the value of the equivalent attribute of the ARL entry.

AR Name

This parameter contains the value of the equivalent attribute of the ARL entry.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.3 Set Cfg

This service is used by the application of the DP-Slave to pass its own configuration data to the DP-Application Layer for the MS0 AR of the DP-Slave. The service description is given in 9.2.5.3.5.

9.3.5.3.4 Load ARL DP Master C11

The service given in Table 274 is used to load the ARL of a DP-Master (Class 1).

Table 274 – Load ARL DP Master C11

Parameter name	.req	.cnf
Argument	M	
Alarm Max	C	
List of ARL Entries	M	
AREP	M	
AR Type	M	
List of related CRL Entries	M	
CREP	M	
Error Action Flag	C	
Min Slave Interval	C	
Data Control Time	C	
Max User Global Control	C	
Max Diag Data Len	C	
Extra Alarm SAP	C	
Max Channel Data Length	C	
Alarm Mode	C	
MS1 Timeout	C	
Poll Timeout	M	
Ident Number	M	
AR Name	O	
Result(+)		S
Result(-)		S
Status		M

Argument

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

Alarm Max

This parameter contains the value for the equivalent attribute of the ARL entry.

List of ARL Entries

This parameter is composed of the following list elements:

AREP

This parameter contains the value for the equivalent attribute of the ARL entry.

AR Type

This parameter contains the value for the equivalent attribute of the ARL entry.

List of related CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value for the equivalent attribute of the ARL entry.

Error Action Flag

This parameter contains the value for the equivalent attribute of the ARL entry.

Min Slave Interval

This parameter contains the value for the equivalent attribute of the ARL entry.

Data Control Time

This parameter contains the value for the equivalent attribute of the ARL entry.

Max User Global Control

This parameter contains the value for the equivalent attribute of the ARL entry.

Max Diag Data Len

This parameter contains the value for the equivalent attribute of the ARL entry.

Extra Alarm SAP

This parameter contains the value for the equivalent attribute of the ARL entry.

Max Channel Data Length

This parameter contains the value for the equivalent attribute of the ARL entry.

Alarm Mode

This parameter contains the value for the equivalent attribute of the ARL entry.

MS1 Timeout

This parameter contains the value for the equivalent attribute of the ARL entry.

Poll Timeout

This parameter contains the value for the equivalent attribute of the ARL entry.

Ident Number

This parameter contains the value for the equivalent attribute of the ARL entry.

AR Name

This parameter contains the value for the equivalent attribute of the ARL entry.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.5 Get ARL DP Master C11

The service given in Table 275 is used to get the ARL of a DP-Master (Class 1).

Table 275 – Get ARL DP Master C11

Parameter name	.req	.cnf
Argument	M	
Result(+)		S
Alarm Max		C
List of ARL Entries		M
AREP		M
AR Type		M
List of related CRL Entries		M
CREP		M
Error Action Flag		C
Min Slave Interval		C
Data Control Time		C
Max User Global Control		C
Max Diag Data Len		C
Extra Alarm SAP		C
Max Channel Data Length		C
Alarm Mode		C
MS1 Timeout		C
Poll Timeout		M
Ident Number		M
AR Name		O
Result(-)		S
Status		M

Argument

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

Result(+)

This selection type parameter indicates that the service request succeeded.

Alarm Max

This parameter contains the value of the equivalent attribute of the ARL entry.

List of ARL Entries

This parameter is composed of the following list elements:

AREP

This parameter contains the value of the equivalent attribute of the ARL entry.

AR Type

This parameter contains the value of the equivalent attribute of the ARL entry.

List of related CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value of the equivalent attribute of the ARL entry.

Error Action Flag

This parameter contains the value of the equivalent attribute of the ARL entry.

Min Slave Interval

This parameter contains the value of the equivalent attribute of the ARL entry.

Data Control Time

This parameter contains the value of the equivalent attribute of the ARL entry.

Max User Global Control

This parameter contains the value of the equivalent attribute of the ARL entry.

Max Diag Data Len

This parameter contains the value of the equivalent attribute of the ARL entry.

Extra Alarm SAP

This parameter contains the value of the equivalent attribute of the ARL entry.

Max Channel Data Length

This parameter contains the value of the equivalent attribute of the ARL entry.

Alarm Mode

This parameter contains the value of the equivalent attribute of the ARL entry.

MS1 Timeout

This parameter contains the value of the equivalent attribute of the ARL entry.

Poll Timeout

This parameter contains the value of the equivalent attribute of the ARL entry.

Ident Number

This parameter contains the value of the equivalent attribute of the ARL entry.

AR Name

This parameter contains the value of the equivalent attribute of the ARL entry.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.6 Load ARL DP Master CI2

The service given in Table 276 is used to load the ARL of a DP-Master (Class 2).

Table 276 – Load ARL DP Master CI2

Parameter name	.req	.cnf
Argument	M	
List of ARL Entries	M	
AREP	M	
AR Type	M	
List of related CRL Entries	M	
CREP	M	
Poll Timeout	M	
AR Name	O	
Result(+)		S
Result(-)		S
Status		M

Argument

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

List of ARL Entries

This parameter is composed of the following list elements:

AREP

This parameter contains the value for the equivalent attribute of the ARL entry.

AR Type

This parameter contains the value for the equivalent attribute of the ARL entry.

List of related CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value for the equivalent attribute of the ARL entry.

Poll Timeout

This parameter contains the value for the equivalent attribute of the ARL entry.

AR Name

This parameter contains the value for the equivalent attribute of the ARL entry.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.7 Get ARL DP Master C12

The service given in Table 277 is used to get the ARL of a DP-Master (Class 2).

Table 277 – Get ARL DP Master C12

Parameter name	.req	.cnf
Argument	M	
Result(+)		S
List of ARL Entries		M
AREP		M
AR Type		M
List of related CRL Entries		M
CREP		M
Poll Timeout		M
AR Name		O
Result(-)		S
Status		M

Argument

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

Result(+)

This selection type parameter indicates that the service request succeeded.

List of ARL Entries

This parameter is composed of the following list elements:

AREP

This parameter contains the value of the equivalent attribute of the ARL entry.

AR Type

This parameter contains the value of the equivalent attribute of the ARL entry.

List of related CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value of the equivalent attribute of the ARL entry.

Poll Timeout

This parameter contains the value of the equivalent attribute of the ARL entry.

AR Name

This parameter contains the value of the equivalent attribute of the ARL entry.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.8 Load CRL DP Slave

The service given in Table 278 is used to load the CRL of a DP-Slave.

Table 278 – Load CRL DP Slave

Parameter name	.req	.cnf
Argument	M	
List of CRL Entries	M	
CREP	M	
CR Type	M	
Related AREP	M	
S_SAP_index	M	
D_addr	M	
Service_activate	M	
Indication_mode	M	
Max_DLSDU_length_req_low	M	
Max_DLSDU_length_req_high	M	
Max_DLSDU_length_ind_low	M	
Max_DLSDU_length_ind_high	M	
CR Name	O	
Result(+)		S
Result(-)		S
Status		M

Argument

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

List of CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value for the equivalent attribute of the CRL entry.

CR Type

This parameter contains the value for the equivalent attribute of the CRL entry.

Related AREP

This parameter contains the value for the equivalent attribute of the CRL entry.

S_SAP_index

This parameter contains the value for the equivalent attribute of the CRL entry.

D_addr

This parameter contains the value for the equivalent attribute of the CRL entry.

Service_activate

This parameter contains the value for the equivalent attribute of the CRL entry.

Indication_mode

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_low

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_high

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_low

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_high

This parameter contains the value for the equivalent attribute of the CRL entry.

CR Name

This parameter contains the value for the equivalent attribute of the CRL entry.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.9 Get CRL DP Slave

The service given in Table 279 is used to get the CRL of a DP-Slave.

Table 279 – Get CRL DP Slave

Parameter name	.req	.cnf
Argument	M	
Result(+)		S
List of CRL Entries		M
CREP		M
CR Type		M
Related AREP		M
S_SAP_index		M
D_addr		M
Service_activate		M
Indication_mode		M
Max_DLSDU_length_req_low		M
Max_DLSDU_length_req_high		M
Max_DLSDU_length_ind_low		M
Max_DLSDU_length_ind_high		M
CR Name		O
Result(-)		S
Status		M

Argument

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

Result(+)

This selection type parameter indicates that the service request succeeded.

List of CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value of the equivalent attribute of the CRL entry.

CR Type

This parameter contains the value of the equivalent attribute of the CRL entry.

Related AREP

This parameter contains the value of the equivalent attribute of the CRL entry.

S_SAP_index

This parameter contains the value of the equivalent attribute of the CRL entry.

D_addr

This parameter contains the value of the equivalent attribute of the CRL entry.

Service_activate

This parameter contains the value of the equivalent attribute of the CRL entry.

Indication_mode

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_low

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_high

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_low

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_high

This parameter contains the value of the equivalent attribute of the CRL entry.

CR Name

This parameter contains the value of the equivalent attribute of the CRL entry.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.10 Load CRL DP Master CI1

The service given in Table 280 is used to load the CRL of a DP-Master (Class 1).

Table 280 – Load CRL DP Master CI1

Parameter name	.req	.cnf
Argument	M	
List of CRL Entries	M	
CREP	M	
CR Type	M	
Related AREP	M	
S_SAP_index	M	
D_SAP_index	M	
D_addr	M	
Service_activate	M	
Role_in_service	M	
Indication_mode	M	
Max_DLSDU_length_req_low	M	
Max_DLSDU_length_req_high	M	
Max_DLSDU_length_ind_low	M	
Max_DLSDU_length_ind_high	M	
Active	M	
New Prm	M	
Prm Data	M	
Lock	M	
Sync	M	
Freeze	M	
WD On	M	
WD Fact 1	M	
WD Fact 2	M	
Min Tsd	M	
Ident Number	M	
Group Ident	M	
DPV1 Enable	C	
Fail Safe	C	
WD Base	C	
Enable Pull Plug Alarm	C	
Enable Process Alarm	C	
Enable Diagnosis Alarm	C	
Enable Manufacturer Specific Alarm	C	
Enable Status Alarm	C	

(continued on next page)

Enable Update Alarm	C	
Check Cfg Mode	C	
Alarm Mode	C	
User Prm Data	O	
Device Related User Parameter	O	
Data Description	O	
List of Module Related User Parameter	O	
Data Description	O	
List of Cfg Data	M	
Cfg Identifier	S	
Type	M	
Format	M	
Consistency	M	
Length	M	
Special Cfg Identifier	S	
Type	M	
Input Length	C	
Input Format	C	
Input Consistency	C	
Output Length	C	
Output Format	C	
Output Consistency	C	
List of Data Description	O	
Data Description	M	
Manufacturer Specific Data	O	
Diag Upd Delay	M	
Fail Safe	M	
DPV1 Supported	M	
NA to Abort	M	
Ignore ACIr	M	
Output Len	M	
Exp Inp Len	M	
Data Transfer Status	M	
System Diagnosis Status	M	
CR Name	O	
Result(+)		S
Result(-)		S
Status		M

Argument

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

List of CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value for the equivalent attribute of the CRL entry.

CR Type

This parameter contains the value for the equivalent attribute of the CRL entry.

Related AREP

This parameter contains the value for the equivalent attribute of the CRL entry.

S_SAP_index

This parameter contains the value for the equivalent attribute of the CRL entry.

D_SAP_index

This parameter contains the value for the equivalent attribute of the CRL entry.

D_addr

This parameter contains the value for the equivalent attribute of the CRL entry.

Service_activate

This parameter contains the value for the equivalent attribute of the CRL entry.

Role_in_service

This parameter contains the value for the equivalent attribute of the CRL entry.

Indication_mode

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_low

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_high

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_low

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_high

This parameter contains the value for the equivalent attribute of the CRL entry.

Active

This parameter contains the value for the equivalent attribute of the CRL entry.

New Prm

This parameter contains the value for the equivalent attribute of the CRL entry.

Prm Data

This parameter is composed of the following subparameters.

Lock

This parameter contains the value for the equivalent attribute of the CRL entry.

Sync

This parameter contains the value for the equivalent attribute of the CRL entry.

Freeze

This parameter contains the value for the equivalent attribute of the CRL entry.

WD On

This parameter contains the value for the equivalent attribute of the CRL entry.

WD Fact 1

This parameter contains the value for the equivalent attribute of the CRL entry.

WD Fact 2

This parameter contains the value for the equivalent attribute of the CRL entry.

Min Tsdr

This parameter contains the value for the equivalent attribute of the CRL entry.

Ident Number

This parameter contains the value for the equivalent attribute of the CRL entry.

Group Ident

This parameter contains the value for the equivalent attribute of the CRL entry.

DPV1 Enable

This parameter contains the value for the equivalent attribute of the CRL entry.

Fail Safe

This parameter contains the value for the equivalent attribute of the CRL entry.

WD Base

This parameter contains the value for the equivalent attribute of the CRL entry.

Enable Pull Plug Alarm

This parameter contains the value for the equivalent attribute of the CRL entry.

Enable Process Alarm

This parameter contains the value for the equivalent attribute of the CRL entry.

Enable Diagnosis Alarm

This parameter contains the value for the equivalent attribute of the CRL entry.

Enable Manufacturer Specific Alarm

This parameter contains the value for the equivalent attribute of the CRL entry.

Enable Status Alarm

This parameter contains the value for the equivalent attribute of the CRL entry.

Enable Update Alarm

This parameter contains the value for the equivalent attribute of the CRL entry.

Check Cfg Mode

This parameter contains the value for the equivalent attribute of the CRL entry.

Alarm Mode

This parameter contains the value for the equivalent attribute of the CRL entry.

User Prm Data

This parameter is composed of the following subparameters.

Device Related User Parameter

This parameter contains the value for the equivalent attribute of the CRL entry.

Data Description

This parameter contains the value for the equivalent attribute of the CRL entry.

List of Module Related User Parameter

This parameter is composed of the following list elements:

Data Description

This parameter contains the value for the equivalent attribute of the CRL entry.

List of Cfg Data

This parameter is composed of the following list elements:

Cfg Identifier

This selection type parameter indicates the type of Cfg Identifier format.

Type

This parameter contains the value for the equivalent attribute of the CRL entry.

Format

This parameter contains the value for the equivalent attribute of the CRL entry.

Consistency

This parameter contains the value for the equivalent attribute of the CRL entry.

Length

This parameter contains the value for the equivalent attribute of the CRL entry.

Special Cfg Identifier

This selection type parameter indicates the type of Cfg Identifier format.

Type

This parameter contains the value for the equivalent attribute of the CRL entry.

Input Length

This parameter contains the value for the equivalent attribute of the CRL entry.

Input Format

This parameter contains the value for the equivalent attribute of the CRL entry.

Input Consistency

This parameter contains the value for the equivalent attribute of the CRL entry.

Output Length

This parameter contains the value for the equivalent attribute of the CRL entry.

Output Format

This parameter contains the value for the equivalent attribute of the CRL entry.

Output Consistency

This parameter contains the value for the equivalent attribute of the CRL entry.

List of Data Description

This parameter is composed of the following list elements:

Data Description

This parameter contains the value for the equivalent attribute of the CRL entry.

Manufacturer Specific Data

This parameter contains the value for the equivalent attribute of the CRL entry.

Diag Upd Delay

This parameter contains the value for the equivalent attribute of the CRL entry.

Fail Safe

This parameter contains the value for the equivalent attribute of the CRL entry.

DPV1 Supported

This parameter contains the value for the equivalent attribute of the CRL entry.

NA to Abort

This parameter contains the value for the equivalent attribute of the CRL entry.

Ignore ACIr

This parameter contains the value for the equivalent attribute of the CRL entry.

Output Len

This parameter contains the value for the equivalent attribute of the CRL entry.

Exp Inp Len

This parameter contains the value for the equivalent attribute of the CRL entry.

Data Transfer Status

This parameter contains the value for the equivalent attribute of the CRL entry.

System Diagnosis Status

This parameter contains the value for the equivalent attribute of the CRL entry.

CR Name

This parameter contains the value for the equivalent attribute of the CRL entry.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.11 Get CRL DP Master C11

The service given in Table 281 is used to get the CRL of a DP-Master (Class 1).

Table 281 – Get CRL DP Master C11

Parameter name	.req	.cnf
Argument	M	
Result(+)		S
List of CRL Entries		M
CREP		M
CR Type		M
Related AREP		M
S_SAP_index		M
D_SAP_index		M
D_addr		M
Service_activate		M
Role_in_service		M
Indication_mode		M
Max_DLSDU_length_req_low		M
Max_DLSDU_length_req_high		M
Max_DLSDU_length_ind_low		M
Max_DLSDU_length_ind_high		M
Active		M
New Prm		M
Prm Data		M
Lock		M
Sync		M
Freeze		M
WD On		M
WD Fact 1		M
WD Fact 2		M
Min Tsdr		M
Ident Number		M
Group Ident		M
DPV1 Enable		C
Fail Safe		C
WD Base		C
Enable Pull Plug Alarm		C
Enable Process Alarm		C
Enable Diagnosis Alarm		C

(continued on next page)

Enable Manufacturer Specific Alarm	C
Enable Status Alarm	C
Enable Update Alarm	C
Check Cfg Mode	C
Alarm Mode	C
User Prm Data	O
Device Related User Parameter	O
Data Description	O
List of Module Related User Parameter	O
Data Description	O
List of Cfg Data	M
Cfg Identifier	S
Type	M
Format	M
Consistency	M
Length	M
Special Cfg Identifier	S
Type	M
Input Length	C
Input Format	C
Input Consistency	C
Output Length	C
Output Format	C
Output Consistency	C
List of Data Description	O
Data Description	M
Manufacturer Specific Data	O
Diag Upd Delay	M
Fail Safe	M
DPV1 Supported	M
NA to Abort	M
Ignore ACIr	M
Output Len	M
Exp Inp Len	M
Data Transfer Status	M
System Diagnosis Status	M
CR Name	O
Result(-)	S
Status	M

Argument

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

Result(+)

This selection type parameter indicates that the service request succeeded.

List of CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value of the equivalent attribute of the CRL entry.

CR Type

This parameter contains the value of the equivalent attribute of the CRL entry.

Related AREP

This parameter contains the value of the equivalent attribute of the CRL entry.

S_SAP_index

This parameter contains the value of the equivalent attribute of the CRL entry.

D_SAP_index

This parameter contains the value of the equivalent attribute of the CRL entry.

D_addr

This parameter contains the value of the equivalent attribute of the CRL entry.

Service_activate

This parameter contains the value of the equivalent attribute of the CRL entry.

Role_in_service

This parameter contains the value of the equivalent attribute of the CRL entry.

Indication_mode

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_low

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_high

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_low

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_high

This parameter contains the value of the equivalent attribute of the CRL entry.

Active

This parameter contains the value of the equivalent attribute of the CRL entry.

New Prm

This parameter contains the value of the equivalent attribute of the CRL entry.

Prm Data

This parameter is composed of the following subparameters.

Lock

This parameter contains the value of the equivalent attribute of the CRL entry.

Sync

This parameter contains the value of the equivalent attribute of the CRL entry.

Freeze

This parameter contains the value of the equivalent attribute of the CRL entry.

WD On

This parameter contains the value of the equivalent attribute of the CRL entry.

WD Fact 1

This parameter contains the value of the equivalent attribute of the CRL entry.

WD Fact 2

This parameter contains the value of the equivalent attribute of the CRL entry.

Min Tsdr

This parameter contains the value of the equivalent attribute of the CRL entry.

Ident Number

This parameter contains the value of the equivalent attribute of the CRL entry.

Group Ident

This parameter contains the value of the equivalent attribute of the CRL entry.

DPV1 Enable

This parameter contains the value of the equivalent attribute of the CRL entry.

Fail Safe

This parameter contains the value of the equivalent attribute of the CRL entry.

WD Base

This parameter contains the value of the equivalent attribute of the CRL entry.

Enable Pull Plug Alarm

This parameter contains the value of the equivalent attribute of the CRL entry.

Enable Process Alarm

This parameter contains the value of the equivalent attribute of the CRL entry.

Enable Diagnosis Alarm

This parameter contains the value of the equivalent attribute of the CRL entry.

Enable Manufacturer Specific Alarm

This parameter contains the value of the equivalent attribute of the CRL entry.

Enable Status Alarm

This parameter contains the value of the equivalent attribute of the CRL entry.

Enable Update Alarm

This parameter contains the value of the equivalent attribute of the CRL entry.

Check Cfg Mode

This parameter contains the value of the equivalent attribute of the CRL entry.

Alarm Mode

This parameter contains the value of the equivalent attribute of the CRL entry.

User Prm Data

This parameter is composed of the following subparameters.

Device Related User Parameter

This parameter contains the value of the equivalent attribute of the CRL entry.

Data Description

This parameter contains the value of the equivalent attribute of the CRL entry.

List of Module Related User Parameter

This parameter is composed of the following list elements:

Data Description

This parameter contains the value of the equivalent attribute of the CRL entry.

List of Cfg Data

This parameter is composed of the following list elements:

Cfg Identifier

This selection type parameter indicates the type of Cfg Identifier format.

Type

This parameter contains the value of the equivalent attribute of the CRL entry.

Format

This parameter contains the value of the equivalent attribute of the CRL entry.

Consistency

This parameter contains the value of the equivalent attribute of the CRL entry.

Length

This parameter contains the value of the equivalent attribute of the CRL entry.

Special Cfg Identifier

This selection type parameter indicates the type of Cfg Identifier format.

Type

This parameter contains the value of the equivalent attribute of the CRL entry.

Input Length

This parameter contains the value of the equivalent attribute of the CRL entry.

Input Format

This parameter contains the value of the equivalent attribute of the CRL entry.

Input Consistency

This parameter contains the value of the equivalent attribute of the CRL entry.

Output Length

This parameter contains the value of the equivalent attribute of the CRL entry.

Output Format

This parameter contains the value of the equivalent attribute of the CRL entry.

Output Consistency

This parameter contains the value of the equivalent attribute of the CRL entry.

List of Data Description

This parameter is composed of the following list elements:

Data Description

This parameter contains the value of the equivalent attribute of the CRL entry.

Manufacturer Specific Data

This parameter contains the value of the equivalent attribute of the CRL entry.

Diag Upd Delay

This parameter contains the value of the equivalent attribute of the CRL entry.

Fail Safe

This parameter contains the value of the equivalent attribute of the CRL entry.

DPV1 Supported

This parameter contains the value of the equivalent attribute of the CRL entry.

NA to Abort

This parameter contains the value of the equivalent attribute of the CRL entry.

Ignore ACIr

This parameter contains the value of the equivalent attribute of the CRL entry.

Output Len

This parameter contains the value of the equivalent attribute of the CRL entry.

Exp Inp Len

This parameter contains the value of the equivalent attribute of the CRL entry.

Data Transfer Status

This parameter contains the value of the equivalent attribute of the CRL entry.

System Diagnosis Status

This parameter contains the value of the equivalent attribute of the CRL entry.

CR Name

This parameter contains the value of the equivalent attribute of the CRL entry.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure

Allowed values: NO, IV

9.3.5.3.12 Load CRL DP Master CI2

The service given in Table 282 is used to load the CRL of a DP-Master (Class 2).

Table 282 – Load CRL DP Master CI2

Parameter name	.req	.cnf
Argument	M	
List of CRL Entries	M	
CREP	M	
CR Type	M	
Related AREP	M	
S_SAP_index	M	
D_SAP_index	M	
D_addr	M	
Service_activate	M	
Role_in_service	M	
Max_DLSDU_length_req_low	M	
Max_DLSDU_length_req_high	M	
Max_DLSDU_length_ind_low	M	
Max_DLSDU_length_ind_high	M	
CR Name	O	
Result(+)		S
Result(-)		S
Status		M

Argument

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

List of CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value for the equivalent attribute of the CRL entry.

CR Type

This parameter contains the value for the equivalent attribute of the CRL entry.

Related AREP

This parameter contains the value for the equivalent attribute of the CRL entry.

S_SAP_index

This parameter contains the value for the equivalent attribute of the CRL entry.

D_SAP_index

This parameter contains the value for the equivalent attribute of the CRL entry.

D_addr

This parameter contains the value for the equivalent attribute of the CRL entry.

Service_activate

This parameter contains the value for the equivalent attribute of the CRL entry.

Role_in_service

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_low

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_high

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_low

This parameter contains the value for the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_high

This parameter contains the value for the equivalent attribute of the CRL entry.

CR Name

This parameter contains the value for the equivalent attribute of the CRL entry.

Result(+)

This selection type parameter indicates that the service request succeeded.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.3.5.3.13 Get CRL DP Master CI2

The service given in Table 283 is used to get the CRL of a DP-Master (Class 2).

Table 283 – Get CRL DP Master CI2

Parameter name	.req	.cnf
Argument	M	
Result(+)		S
List of CRL Entries		M
CREP		M
CR Type		M
Related AREP		M
S_SAP_index		M
D_SAP_index		M
D_addr		M
Service_activate		M
Role_in_service		M
Max_DLSDU_length_req_low		M
Max_DLSDU_length_req_high		M
Max_DLSDU_length_ind_low		M
Max_DLSDU_length_ind_high		M
CR Name		O
Result(-)		S
Status		M

Argument

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

Result(+)

This selection type parameter indicates that the service request succeeded.

List of CRL Entries

This parameter is composed of the following list elements:

CREP

This parameter contains the value of the equivalent attribute of the CRL entry.

CR Type

This parameter contains the value of the equivalent attribute of the CRL entry.

Related AREP

This parameter contains the value of the equivalent attribute of the CRL entry.

S_SAP_index

This parameter contains the value of the equivalent attribute of the CRL entry.

D_SAP_index

This parameter contains the value of the equivalent attribute of the CRL entry.

D_addr

This parameter contains the value of the equivalent attribute of the CRL entry.

Service_activate

This parameter contains the value of the equivalent attribute of the CRL entry.

Role_in_service

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_low

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_req_high

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_low

This parameter contains the value of the equivalent attribute of the CRL entry.

Max_DLSDU_length_ind_high

This parameter contains the value of the equivalent attribute of the CRL entry.

CR Name

This parameter contains the value of the equivalent attribute of the CRL entry.

Result(-)

This selection type parameter indicates that the service request failed.

Status

This parameter indicates the reason for the failure.

Allowed values: NO, IV

9.4 Summary of Type 3 FAL Classes

This subclause contains a summary of the defined Type 3 FAL Classes.

Table 284 – Type 3 FAL Class Summary

FAL ASE	Class
Process Data	Simple Pocess Data Combined Pocess Data
I/O Data	Simple Input Data Simple Output Data Extended Input Data Extended Output Data
Diagnosis	Device Related Diagnosis Identifier Related Diagnosis Channel Related Diagnosis Status Module Status
Alarm	Alarm
Context	MS0 User Parameter MS0 Configuration Elements Remanent Parameter MS2 User Parameter
Management	Master Diag Master Parameter
Application Relationship List	ARL DP-Slave ARL DP-Master (Class 1) ARL DP-Master (Class 2)
Communication Relationship List	CRL DP-Slave CRL DP-Master (Class 1) CRL DP-Master (Class 2)

9.5 Permitted FAL Type 3 Services by AREP Role

Table 285 gives an overview of the requester and responder roles in service execution. Some services are additionally possible. Requester and responder roles in service execution are given separately.

Table 285 – Assignment of the services to DP-Masters and DP-Slaves

Service	DP-Slave			DP-Master (Class 1)			DP-Master (Class 2)		
	MS0	MS1	MS2	MS0	MS1	MM1/MM2	MS2	MS0	MM1/MM2
Read		R	R		I		I		
Write		R	R		I		I		
Data Transport			R				I		
Set Input	LR								
Read Input	(R)							I	
Get Input				LR					
New Input				LI					
Set Output				LR					
Read Output	(R)							I	
Get Output	LR								
New Output	LI								
Global Control	R			I					
Set Slave Diag	LR								
Get Slave Diag				LR					
Read Slave Diag	(R)							I	
New Slave Diag				LI					
Alarm Notification		I			R				
Alarm Ack		R			I				
Initiate			R				I		
Abort			R/I				I/R		
Set Prm	(R)			I					
Check User Prm	LI								
Check User Prm Result	LR								
Set Cfg	LR								
Get Cfg	(R)							I	
Check Cfg	LI								
Check Cfg Result	LR								
Set Slave Add	R							I	
Get Master Diag						R			I
Start Seq						R			I
Download						R			I
Upload						R			I
End Seq						R			I
Act Para Brct						R			I
Act Param						R			I

where

- I Initiator of a Service
- R Receiver (Responder) of a Service
- (R) Receiver (but no interaction with the application)
- LR Local service initiated by the application
- LI Local service initiated by the DP-Application Layer

Table 286 shows the support of ARs in the different DP-Device Types.

Table 286 – Support of AR Types in the different DP-Device Types

AR Type	DP-Slave	DP-Master (Class 1)	DP-Master (Class 2)
MS0	M	M	O
MS1	O	O	--
MS2	O	--	O
MM1	--	M	O
MM2	--	O	O
where M AR Type has to be supported mandatorily O AR Type is supported optionally -- AR Type not supported			

Table 287 gives an overview of the services which shall be supported at the different AREPs respectively CREPs.

Table 287 – Support of Services at the different AREPs respectively CREPs

AR Type	CR Type	CR Name	Supported Service at the AREP/ CREP		
			DP-Slave	DP-Master (Class 1)	DP-Master (Class 2)
MS0	BBUB	I/O	Set Input (M)	--	--
			Get Output (M)	--	--
			New Output (O)	--	--
			--	Set Output (M)	--
			--	Get Input (M)	--
			--	New Input (O)	--
	BQUU	Cfg	Set Cfg (M)	--	--
			Check Cfg (M)	--	--
			Check Cfg Result (M)	--	--
	BQUU	Prm	--	Set Prm (M)	--
			Check User Prm (M)	--	--
			Check User Prm Result (M)	--	--
	BBUU	Diag	Set Slave Diag (M)	--	--
			--	Get Slave Diag (M)	--
			--	New Slave Diag (O)	--
	QQUU-OM	I/O GC	Global Control (M)	Global Control (M)	--
	QBCB	Read Diag	Read Slave Diag (M)	--	Read Slave Diag (O)
	QBCB	Read Cfg	Get Cfg (M)	--	Get Cfg (O)
	QBCB	Read Output	Read Output (M)	--	Read Output (O)
	QBCB	Read Input	Read Input (M)	--	Read Input (O)
	QBCB	Rem Para	Set Slave Add (O)	--	Set Slave Add (O)
MS1	QQOB-CO	Process-Data Alarm Ack	Read (O)	Read (O)	--
			Write (O)	Write (O)	--
			Alarm Ack (O)	Alarm Ack (O)	--
	QQOB-CO	Alarm Ack	Alarm Ack (O)	Alarm Ack (O)	--
MS2	QQUU-OO	Context MS2	Initiate (O)	--	Initiate (O)
			Abort (O)	--	Abort (O)
			--	--	--
	QQCB-CO	Process-Data	Read (O)	--	Read (O)
			Write (O)	--	Write (O)
			Data Transport (O)	--	Data Transport (O)
MM1	QBCB-CL	MM	--	Get Master Diag (M)	Get Master Diag (O)
			--	Start Seq (O)	Start Seq (O)
			--	Download (O)	Download (O)
			--	Upload (O)	Upload (O)
			--	End Seq (O)	End Seq (O)
			--	Act Param (O)	Act Param (O)
MM2	QQUU-OM	Act Para Brct	--	Act Para Brct (O)	Act Para Brct (O)
where			mandatory service at this AREP / CREP		
M			optional service at this AREP / CREP		
O			non-existent service at this AREP / CREP		
--					

9.6 Conformance Classes

Conformance Classes are used to define the minimum functionality of a DP-Master. There are no Conformance Classes for DP-Slaves. The functionality of the DP-Slave determines the necessary conformance class of the DP-Master.

For the cyclic data exchange the DP-Master (Class 1) has to support the following features:

- min. 2 words consistency
- min. 32 byte data length
- new DPV1 data types except Date, Time-Of-Day, Time-Difference
- Fail Safe has to be supported
- min. 32 byte Diag Data

For the acyclic communication the DP-Master has to support the maximum data length of 244 bytes.

Table 288 – Conformance Classes DP-Master (Class 1)

Conformance Class	Functionality MS1
A	Read Write
B	Read Write Alarm Notification Alarm Ack

Table 289 – Conformance Classes DP-Master (Class 2)

Conformance Class	Functionality MS2
A	Read Write Initiate Abort Data Transport

9.7 Application Characteristics

9.7.1 Time Behaviour

The following diagram shows the theoretical system reaction time (see clause "System Reaction Time") of a DP-Master (Class 1) in dependence of the connected amount of DP-Slaves with 2 bytes input- and output-data each.

It is presumed that all configured DP-Slaves are in the data exchange mode with no diagnosis message pending. The DP-Slaves answer immediately after the minimum Station Delay Timer (min $T_{SDR} = 11 t_{Bit}$) is expired. At the DP-Master (Class 1) T_{SDI} is assumed as 37 bit times.

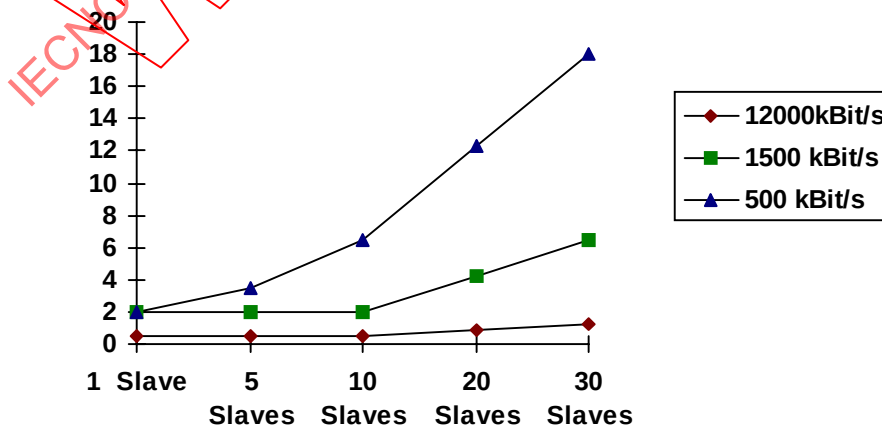


Figure 65 – Cycle time of the DP-system

9.7.2 Ident Number

The Ident Number is necessary for all DP-devices except for DP-Master (Class 2). For each type of device an individual Ident Number will be needed. This Ident Number is not a series number. If a manufacturer have got a Ident Number for one type of device it is allowed to him to use this number for all produced devices of the same type. At devices of the same type but with differences in the amount of inputs and outputs it is possible to use the same Ident Number for each produced DP-device. Assumption: The device can be described in the DDB as a modular device.

Each device type shall have an individual Ident Number. This Ident Number will be issued by the PI.

10 Communication Model Type 4 Specification

10.1 Type 4 Concepts

10.1.1 Overview

The concepts of the Type 4 model basically follow the common concepts with a few exceptions. The basic principle described in the common Concepts clause until 4.2.5 are applicable for this clause. Type 4 supports Gateway functionality within the application layer. Part of this can also be recognised from the common concepts. However, to promote readability, some elements from the common concepts that are relevant for Type 4 are repeated in this clause.

10.1.2 Application Entities

10.1.2.1 Definition of FAL AE

An FAL AE provides a set of services to enable communications between APs in a fieldbus environment. These services enable a user application in a Client to access user objects in a server.

The functionality of the FAL AE can be divided into two groups.

Application Relation (AR). The AR functionality is responsible for arranging the transportation of APDUs via the network.

Variable Access (VA). The Variable Access functionality is responsible for coding and encoding of APDUs. In a server the VA also convey data directly to / from user objects. How this is done is a local matter and out of scope for this standard.

10.1.2.2 FAL Services

The services offered by the FAL AE to the user application are called FAL Services. Services for the AR are called ASE services and services for the Variable Access are called Route Endpoint ASE services (REP ASE services).

The FAL services provide access to local AR objects, and provide conveyance of requests and responses for access to real objects modelled as FAL accessible objects. The FAL provides as well confirmed as unconfirmed services. Service requests are conveyed to the AP containing the real object.

Figure 66 illustrates the Type 4 FAL AE and the architectural relationships.

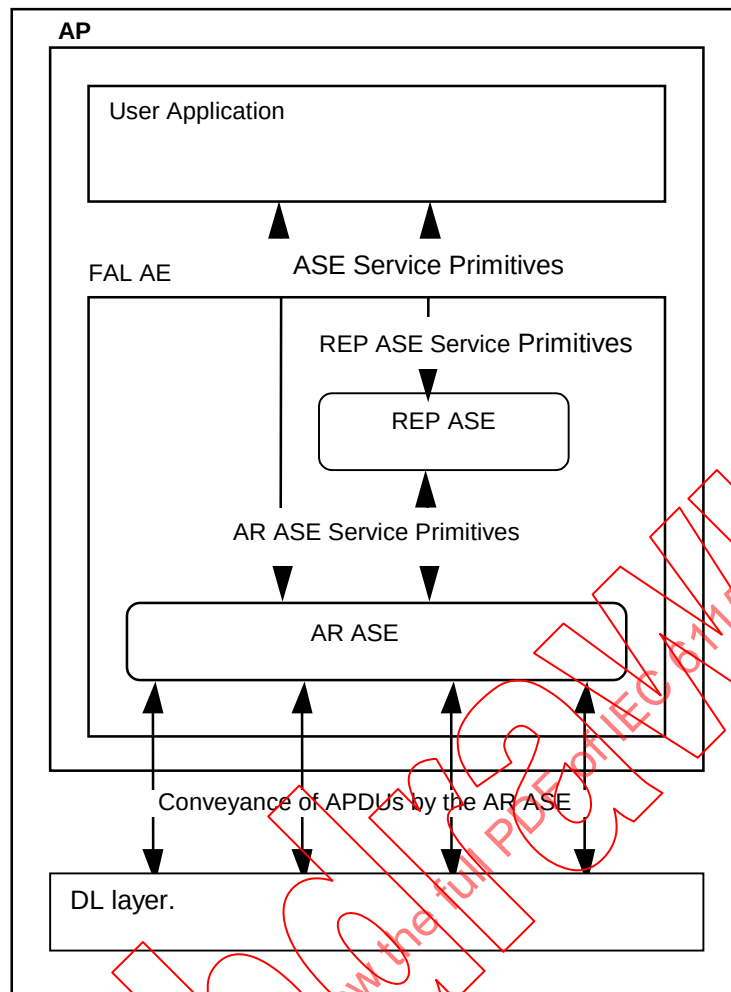


Figure 66 – Type 4 FAL AE

10.1.2.3 AR ASE

To support access to the remote AP, the Application Relationship ASE is defined. It provides services to the AP for accessing communication-related parameters, and it provides services to the REP ASE for conveying service requests and responses.

The AR ASE provides services at the endpoints of ARs (AREPs).

10.1.2.4 REP ASE

The REP ASE provides a set of services used to read and write network visible attributes of Variable Objects and their application data. As only one ASE type is used remotely, the type is implicit and not transferred.

10.1.2.5 FAL Service Conveyance and Gateway

The FAL AE provides services to convey requests and responses between User Applications and Variable Objects including Gateway functionality.

In a distributed system, application processes communicate with each other by exchanging application layer messages. The AR is defined to support the on-demand exchange of confirmed and unconfirmed services between two application processes. Connectionless data link services are used for the exchanges. The data link layer transfers may be either acknowledged or unacknowledged.

10.1.2.6 Application Relationships

10.1.2.6.1 Definition of AR

The AR is responsible for conveying messages between applications. The messages that are conveyed are FAL service requests and responses. Each service request and response is submitted to the AR ASE for transfer. Within the AR only one object type exist, the AR-Endpoint (AREP).

10.1.2.6.2 AR-Endpoints

An AR-Endpoint (AREP) represents a service access point to a DLE within the AE. The AR Send Service conveys APDU and route information to the AREP and implicit to the DLE. The AREP is a system management object presenting the user configurable parameters of the related DLE.

In type 4, there are no special system management services, because the attributes of the AREP objects are declared as Variable Objects by the user application. This makes the attributes network visible. The AR Get Attribute and AR Set Attribute services are used by these Variable Objects to access the attribute values.

When an AREP receives an APDU from its DLE (User-triggered) it conveys the APDU to the REP ASE or an AREP (gateway).

10.1.3 Gateway and Routing

Type 4 supports gateway functionality. To identify the destination of a request, a route description is used. This route description holding a complete list of points to pass is sent with the request. On its way to the destination, the route evolves in such a way that it always contains the route back to the requester and the route to the destination. The elements in the route are endpoint and DL addresses.

10.1.3.1 Route Endpoint

Within the REP ASE, Route Endpoint objects (REPs) represent endpoints, holding a Route description. An REP represents the starting point for conveying a message. An REP also represents the destination of the APDU.

10.1.3.2 Identification of Endpoints

The endpoints are identified by endpoint addresses. These EP addresses are numerical Addresses, taken within an AE from the same name space of numbers, to achieve a unique identification. These EP addresses are used to identify AREPs and REPs.

10.1.3.3 DL addresses

The DLE is identified from the Link by a DL-address. DL-addresses are defined in IEC 61158-4, but as they are part of the route, they are mentioned here. There is a one to one relation between the DL-address of a DLE and the endpoint address of the corresponding AREP.

10.1.3.4 Route

The complete Route contains a Destination-Route and a Source-Route.

The Destination-Route describes how to reach the destination REP. It is a sequence of Endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the receiving DLE, AREP or REP.

The Source Route in the same way describes how to find the way back to the endpoint that initiated the request. It is a sequence of endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the transmitting DLE, AREP or REP.

10.1.3.5 Destination /Source Route conversion.

A Destination-Route conveyed by an REP is starting with the endpoint address of the local AREP. The next element is the DL-address of the receiving DLE. If the route goes through gateways a pair of addresses must be added for each gateway, a DL-address and an AREP address. The final address is the endpoint address of the destination REP. The Source address is the endpoint address of the source REP.

The AR Send service removes the first element of the Destination-Route. The DLE inserts its own DL-address in front of the current Source Route before conveying the request to the link.

When a DLE receives an indication from the link, it removes the first element of the Destination Route (its own DL-address). The corresponding AREP inserts its own AREP address in front of the current Source Route before conveying the request to a REP or an AREP.

At the destination REP the Destination-Route holds the endpoint address of the REP, and the Source Route holds the complete route back to the requesting REP.

If any error appears with a request on its way to the destination, the current Source Route can be used for returning an error response.

10.1.4 Architecture Summary

A summary of the FAL architecture is presented in Figure 67. It illustrates the major components of the FAL architecture and how they relate to each other.

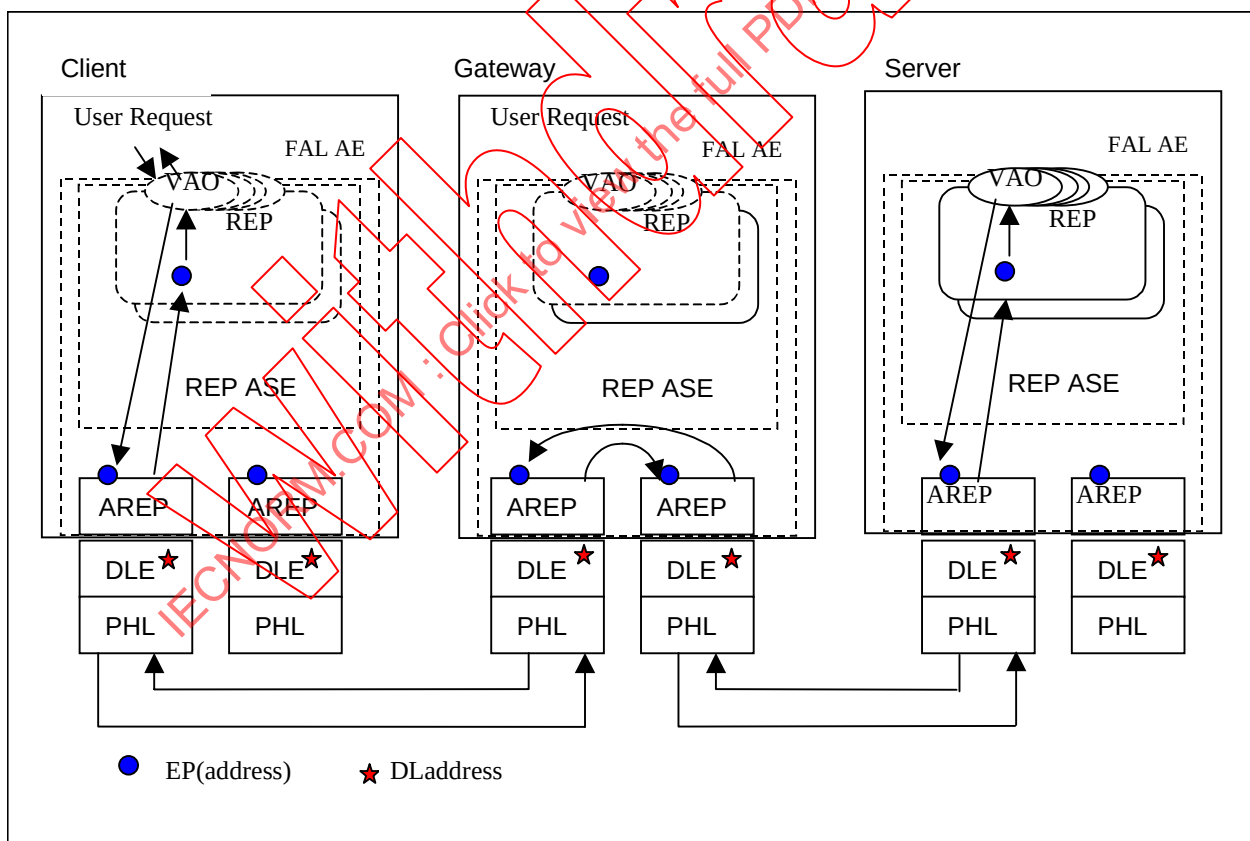


Figure 67 – Summary of the Type 4 FAL architecture

10.1.5 FAL Service Procedures and Time Sequence Diagrams

This subclause describes service procedures for FAL confirmed and unconfirmed services. An overview of the FAL service procedures is given in Figure 68.

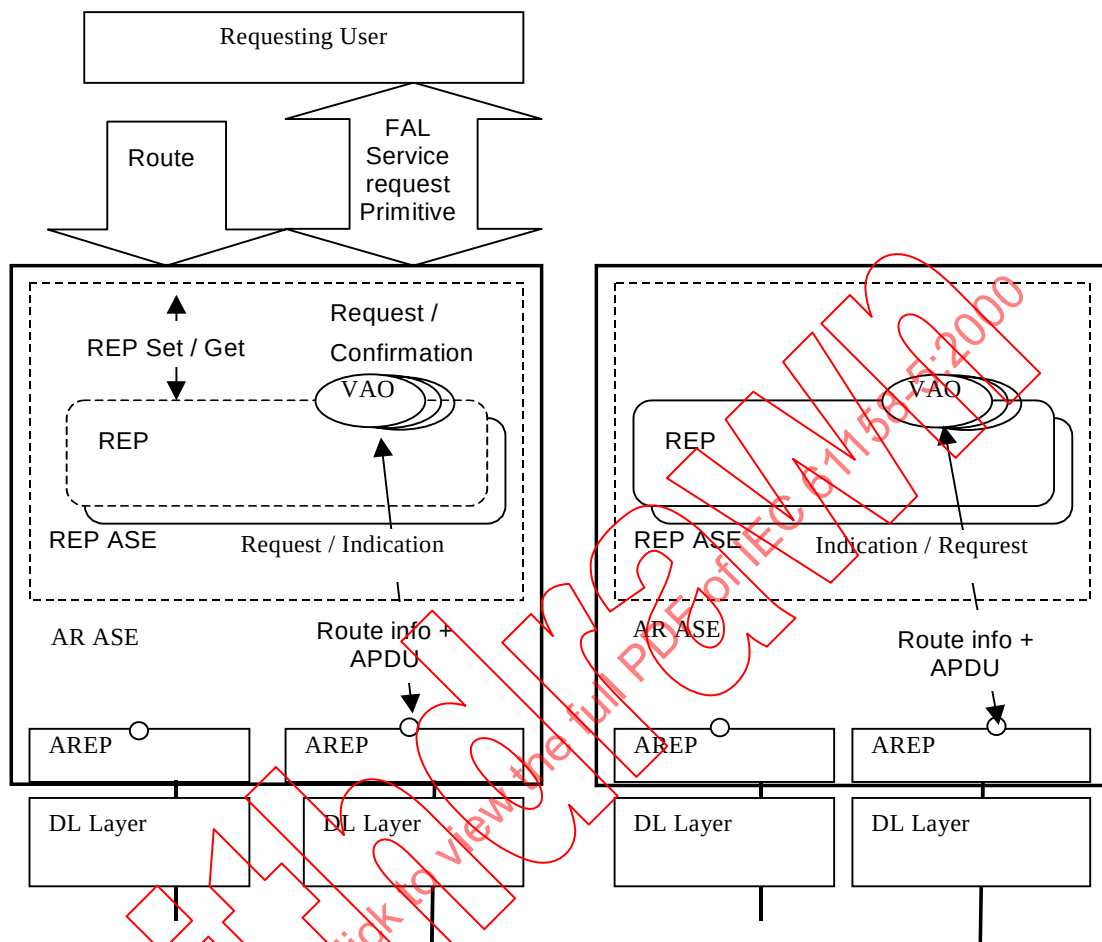


Figure 68 – FAL service procedure overview

10.1.5.1 FAL Confirmed Service Procedures

At requesting AP:

The user application uses the REP ASE service "Reserve REP" to reserve a REP. The user application then uses the REP ASE service "Set REP Attribute" to set attributes of the REP, including the Destination Route to the remote REP.

The requesting user submits a confirmed request primitive to the REP ASE with the REP address as a parameter. The REP ASE builds the related confirmed request APDU and conveys it on the AREP, using an AR Send request primitive. The AREP address is indicated by the first element of the Destination route. The REP ASE starts an associated timer to monitor the request.

The AREP builds a request DLSDU and submits the route info together with the DLSDU to its DLE. The DLE sends the DLSDU at the first opportunity.

At responding server AP:

Upon receipt of the DLSDU, the receiving AREP builds an APDU and delivers the APDU and the received Route to the addressed REP by an AR Send indication primitive.

The REP ASE handles the indication by data exchange with the addressed Variable Object, builds the related unconfirmed response APDU and conveys it on the AREP, using an AR Send request primitive. The received Source route is used as Destination route for the response.

The AREP builds a request DLSDU and submits the route info, together with the DLSDU to its DLE. It is the task of the DL layer to decide whether to “queue” the DLSDU or send it immediately.

If the REP ASE is not able to process the indication and respond within “MaxIndicationDelay”, the REP ASE must submit an AR Acknowledge request primitive to the receiving AREP, and in this way free the Token. The AR Acknowledge is submitted to the DLE.

At requesting AP receiving the response:

Upon receipt of the response DLSDU, the receiving AREP conveys the response to the REP ASE. The destination address conveyed to the REP ASE holds the endpoint address of the requesting REP. The REP object can now deliver the response to the requesting user application. This is accomplished by the user invocation of the REP ASE RESPONSE Service.

If the timer expires before the requesting REP has received the response, the REP cancels the associated transaction state machine and delivers the Error response “No answer” to the user application.

10.1.5.2 Confirmed Service Time Sequence Diagram

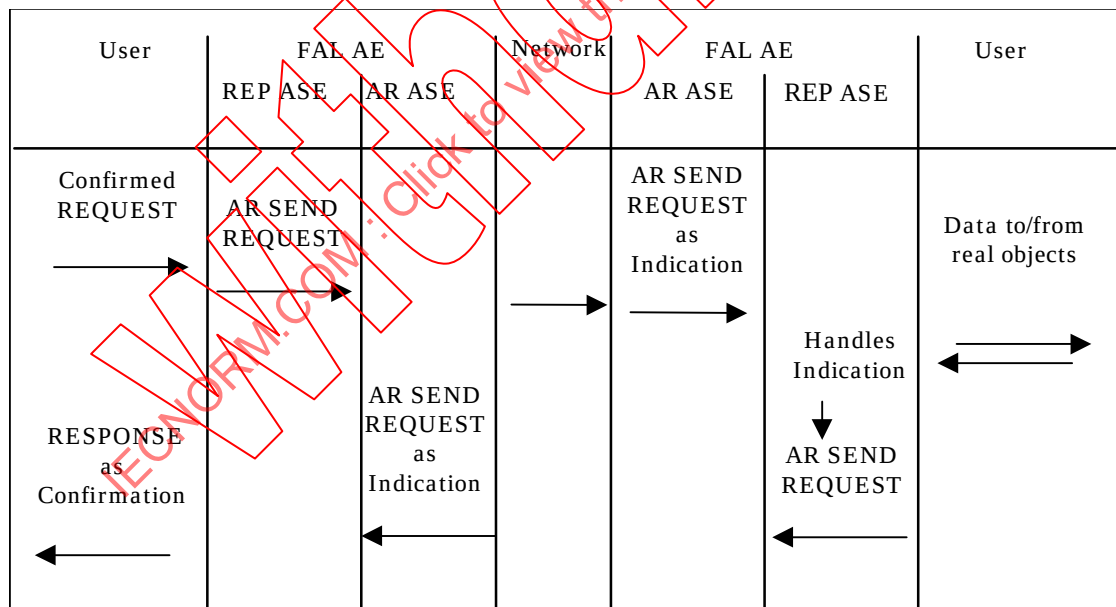


Figure 69 – Time Sequence Diagram for the Confirmed Services

10.1.5.3 Unconfirmed Request Service Procedure

At requesting AP:

The user application uses the REP ASE service "Reserve REP" to reserve a REP. The user application then uses the REP ASE service "Set REP Attribute" to set attributes of the REP, including the Destination Route to the remote REP.

The requesting user submits an unconfirmed request primitive to the REP ASE with the REP address as a parameter. The REP ASE builds the related unconfirmed request APDU and conveys it on the AREP, using an AR Send request primitive. The AREP address is indicated by the first element of the Destination route.

The AREP builds a request DLSDU and submits the route info together with the DLSDU to its DLE. The DLE sends the DLSDU at the first opportunity.

At responding server AP:

Upon receipt of the DLSDU, the receiving AREP builds an APDU and delivers that and the received Route to the addressed REP by an AR Send indication primitive.

The REP ASE handles the indication by data exchange with the addressed Variable Object.

10.1.5.4 Unconfirmed Service Time Sequence Diagram

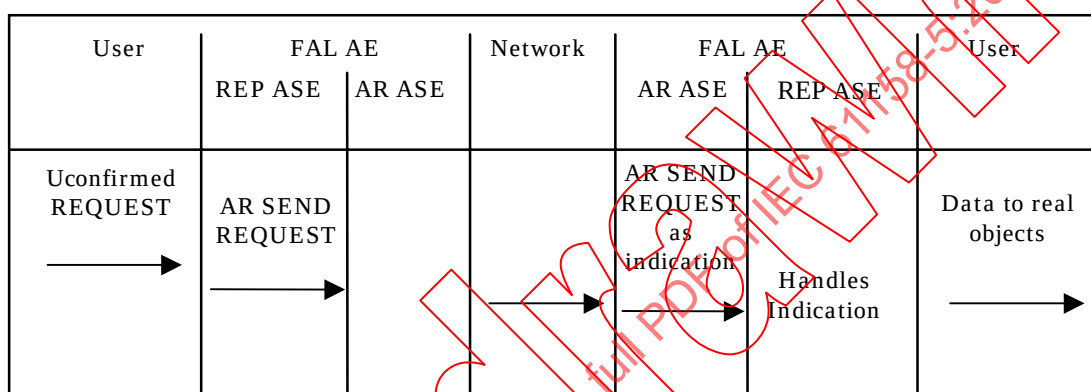


Figure 70 – Time Sequence Diagram for Unconfirmed Services

10.1.5.5 Confirmed / Unconfirmed Service Procedure through gateways

Upon receipt of a DLSDU, the receiving AREP delivers an AR Send indication primitive to the local destination. If the endpoint address is a REP address, the APDU + Route is delivered to the REP ASE. If the endpoint address is an AREP address, the AR send service is used to pass it on to the network (Gateway).

For confirmed service Request, an AR Acknowledge is submitted to the “source AREP” unless the response is expected within “MaxIndicationDelay”. “MaxIndicationDelay” is an attribute of the “source AREP”.

10.2 Type 4 Variable ASE

10.2.1 Overview

The type 4 Variable ASE specifies the machine independent syntax for application data conveyed by FAL services. The application layer supports the transfer of both basic and constructed variables. The encoding rules for basic variables and constructed variables as specified in this subclause are provided in IEC 61158-6.

In type 4, Variable Objects (VAOs) are defined as instances of a variable class. The Variable ASE provides access to VAOs. VAOs are identified by a numeric Variable Identifier, and are instances of a variable class.

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 free.

Constructed types are packed according to the encoding rules for constructed types. The structure of a constructed type is not network visible. The structure of a constructed type is generated by a compiler and transferred by other means.

All variable classes have the common attributes Variable Identifier, Data Type Identifier, Octet length, Read enable, Write enable and Write protected. All variable classes have the common services Read, Write, Get Variable Attribute and Set Variable Attribute.

This standard recommends using floating point representation of measurement values scaled in SI units.

10.2.1.1 Overview of Basic Variable Types

Basic variable types have a constant length. The length of basic variable types is an integral number of octets. Defining new basic variable types is accomplished by defining new basic variable type classes.

10.2.1.2 Overview of Constructed Variable Types

Constructed variable types are needed to completely convey the variety of information present on the Fieldbus. Four kinds of constructed variable types are defined in four classes in this standard: Complex, String, BitString and FIFO. Defining new kinds of constructed variable types is accomplished by defining new constructed variable type classes.

On the Fieldbus a complete constructed variable can be transferred, or a part of it, e.g. a single field of a complex variable or an element of a string. When transferring only part of a variable, addressing with offset and maybe Bit-no is used. Offset indicates the distance (in octets) from the first octet of the variable to the part to be transferred, Bit-no indicates the bit number in one octet.

10.2.1.2.1 Complex variable type

A Complex variable type defines Arrays and Structures.

An Array is composed of an ordered set of homogeneously typed elements. This standard places no restriction on the 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.

A Structure is made of an ordered set of heterogeneously typed elements called fields. Like Arrays, this standard does not restrict the type of fields. However, the fields within a Structure can be of different types.

10.2.1.2.2 String variable type

A String is composed of an ordered set of an Actual Number of Elements field and a number of homogeneously typed elements. Once defined, the maximum number of elements in a String may not be changed.

10.2.1.2.3 BitString variable type

A BitString is defined as a series of bits. Once defined, the number of elements in a BitString may not be changed. The octet length of a BitString is the integral part of ((the number of elements + 7) divided by 8).

10.2.1.2.4 FIFO variable type

A FIFO (First In First Out) is composed of a set of homogeneously typed elements. This standard places no restriction on the type of FIFO elements, but it does require that each element be of the same type. The first written element will be the first element that can be read. On the Fieldbus only one, complete element can be transferred as a result of one service invocation.

10.2.2 Variable Model Class Specification

10.2.2.1 Variable Formal Model

The Variable formal model defines the characteristics common to all Variable classes. This class is not capable of being instantiated. It is present only for the inheritance of its attributes and services of its subclasses.

FAL ASE:	VARIABLE ASE
CLASS:	VARIABLE
CLASS ID:	1
PARENT CLASS:	TOP
ATTRIBUTES:	
1 (m) Key Attribute:	Variable Identifier
2 (o) Attribute:	VariableTypeIdentifier
3 (o) Attribute:	Octet Length
4 (o) Attribute:	Read enable
5 (o) Attribute:	Write enable
6 (o) Attribute:	Write protected
SERVICES:	
1 (m) OpsService:	Read
2 (m) OpsService:	Write
3 (o) OpsService:	Get Variable Attribute
4 (o) OpsService:	Set Variable Attribute

10.2.2.1.1 Attributes

Variable Identifier

This attribute identifies the Variable Object instantiating the variable type class.

VariableTypeIdentifier

This optional attribute identifies the variable type as a Boolean, Integer8, Integer16, Integer32, Unsigned8, Unsigned16, Float32, Float64, UNICODE Char, Array, String, Structure or FIFO. The VariableTypeIdentifier is a numerical identifier, defined in IEC 61158-6. In this standard, a descriptive name corresponding to the identifier is stated.

Octet Length

This optional attribute indicates the length in octets of the data of the Variable Object. For all types except FIFO octet length indicates the total number of octets. For FIFO types, octet length indicates the number of octets in one element.

Read enable

The value of this optional attribute indicates, whether the data value of the Variable Object can be read via the Fieldbus.

Write enable

The value of this optional attribute indicates, whether the data value of the Variable Object can be updated via the Fieldbus.

Write protected

The value of this optional attribute indicates, whether the data value of the Variable Object can only be updated via the Fieldbus under certain conditions. The conditions for permitting update of a Write protected VAO are out of scope of this standard.

10.2.2.1.2 Services

Read

This service is used to read the data value of the Variable Object.

Write

This service is used to update the data value of the Variable Object.

Get Variable Attribute

This service is used to read an attribute of the Variable Object.

Set Variable Attribute

This service is used to update an attribute of the Variable Object.

10.2.3 Basic Variable Type Specifications

10.2.3.1 Boolean Variable Type Formal Model

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

FAL ASE:	VARIABLE ASE
CLASS:	BOOLEAN VARIABLE TYPE
CLASS ID:	2
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	
1 (m) OpService:	And
2 (m) OpService:	Or
3 (m) OpService:	Test-And-Set

10.2.3.1.1 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Boolean
3	Octet Length	= 1

10.2.3.1.2 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation.

10.2.3.2 Integer8 Variable Type Formal Model

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

FAL ASE:	VARIABLE ASE
CLASS:	INTEGER8 VARIABLE TYPE
CLASS ID:	3
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	
1	(m) OpsService: And
2	(m) OpsService: Or
3	(m) OpsService: Test-And-Set

10.2.3.2.1 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Integer8
3	Octet Length	= 1

10.2.3.2.2 Services**And**

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation.

10.2.3.3 Integer16 Variable Type Formal Model

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

FAL ASE:	VARIABLE ASE
CLASS:	INTEGER16 VARIABLE TYPE
CLASS ID:	4
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	
1	(m) OpsService: And
2	(m) OpsService: Or

10.2.3.3.1 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Integer16
3	Octet Length	= 2

10.2.3.3.2 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

10.2.3.4 Integer32 Variable Type Formal Model

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

FAL ASE:	VARIABLE ASE
CLASS:	INTEGER32 VARIABLE TYPE
CLASS ID:	5
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	
1 (m) OpService:	And
2 (m) OpService:	Or

10.2.3.4.1 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Integer32
3	Octet Length	= 4

10.2.3.4.2 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

10.2.3.5 Unsigned8 Variable Type Formal Model

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

FAL ASE:	VARIABLE ASE
CLASS:	UNSIGNED8 VARIABLE TYPE
CLASS ID:	6
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	
1 (m) OpsService:	And
2 (m) OpsService:	Or
3 (m) OpsService:	Test-And-Set

10.2.3.5.1 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Unsigned8
3	Octet Length	= 1

10.2.3.5.2 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation.

10.2.3.6 Unsigned16 Variable Type Formal Model

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

FAL ASE:	VARIABLE ASE
CLASS:	UNSIGNED16 VARIABLE TYPE
CLASS ID:	7
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	
1 (m) OpsService:	And
2 (m) OpsService:	Or

10.2.3.6.1 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Unsigned16
3	Octet Length	= 2

10.2.3.6.2 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation.

10.2.3.7 Float32 Variable Type Formal Model

This type has a length of four octets. The format for Float32 is that defined by ANSI/IEEE 754 as single precision.

FAL ASE:	VARIABLE ASE
CLASS:	FLOAT32 VARIABLE TYPE
CLASS ID:	8
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	

10.2.3.7.1 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Float32
3	Octet Length	= 4

10.2.3.8 Float64 Variable Type Formal Model

This type has a length of eight octets. The format for Float64 is that defined by ANSI/IEEE 754 as double precision.

FAL ASE:	VARIABLE ASE
CLASS:	FLOAT64 VARIABLE TYPE
CLASS ID:	9
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	

10.2.3.8.1 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= Float64
3	Octet Length	= 8

10.2.3.9 UNICODE Char Variable Type Formal Model

This type has a length of two octets. It is defined as a single UNICODE character.

FAL ASE:	VARIABLE ASE
CLASS:	UNICODE CHAR VARIABLE TYPE
CLASS ID:	10
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	

10.2.3.9.1 Attributes

Variable Objects of this class will have the following constant values of the following (parent class) attributes:

2	VariableTypeIdentifier	= UNICODE Char
3	Octet Length	= 2

10.2.4 Constructed Variable Type Specifications**10.2.4.1 Complex Variable Type Formal Model**

A Complex Variable is an Array or a Structure.

An Array is composed of an ordered set of homogeneously typed elements. This standard places no restriction on the 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.

A Structure is made of an ordered set of heterogeneously typed elements called fields. Like Arrays, this standard does not restrict the type of fields. However, the fields within a Structure can be of different types.

FAL ASE:	VARIABLE ASE
CLASS:	COMPLEX VARIABLE TYPE
CLASS ID:	11
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	

1	(m)	OpsService:	And
2	(m)	OpsService:	Or
3	(m)	OpsService:	Test-And-Set

10.2.4.1.1 Attributes

Variable Objects of this class will have the following constant value of the following (parent class) attribute:

2 VariableTypeIdentifier = Complex

10.2.4.1.2 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation. The operation is legal on subelements of integer or boolean type only.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation. The operation is legal on subelements of integer or boolean type only.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation. The operation is legal on subelements of Integer8, Unsigned8 or boolean type only.

10.2.4.2 String Variable Type Formal Model

A String is composed of an ordered set of an Actual Number of Elements field and a number of homogeneously typed elements. Once defined, the maximum number of elements in a String may not be changed.

FAL ASE:	VARIABLE ASE
CLASS:	STRING VARIABLE TYPE
CLASS ID:	12
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	
1 (m) OpsService:	And
2 (m) OpsService:	Or
3 (m) OpsService:	Test-And-Set

10.2.4.2.1 Attributes

Variable Objects of this class will have the following constant value of the following (parent class) attribute:

2 VariableTypeIdentifier = String

10.2.4.2.2 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation. The operation is legal on subelements of integer or boolean type only.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation. The operation is legal on subelements of integer or boolean type only.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation. The operation is legal on subelements of Integer8, Unsigned8 or boolean type only.

10.2.4.3 BitString Variable Type Formal Model

A BitString is defined as a series of bits. Once defined, the number of bits in a BitString may not be changed. The octet length of a BitString is the integral part of ((the number of bits + 7) divided by 8).

FAL ASE:	VARIABLE ASE
CLASS:	BITSTRING VARIABLE TYPE
CLASS ID:	13
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
SERVICES:	
1 (m) OpsService:	And
2 (m) OpsService:	Or
3 (m) OpsService:	Test-And-Set

10.2.4.3.1 Attributes

Variable Objects of this class will have the following constant value of the following (parent class) attribute:

2 VariableTypeIdIdentifier = BitString

10.2.4.3.2 Services

And

This service is used to update the data value of the Variable Object, by storing the result of a logical AND operation. The operation is legal on a subelement (which is a bit) only.

Or

This service is used to update the data value of the Variable Object, by storing the result of a logical OR operation. The operation is legal on a subelement (which is a bit) only.

Test-And-Set

This service is used to read and update the data value of the Variable Object, by returning the result of a logical AND operation, and storing the result of a logical OR operation. The operation is legal on a subelement (which is a bit) only.

10.2.4.4 FIFO Variable Type Formal Model

A FIFO (First In First Out) is composed of a set of homogeneously typed elements. This standard places no restriction on the type of FIFO elements, but it does require that each element be of the same type. The first written element will be the first element that can be read. On the Fieldbus only one, complete element can be transferred as a result of one service invocation.

FAL ASE:	VARIABLE ASE
CLASS:	FIFO VARIABLE TYPE
CLASS ID:	14
PARENT CLASS:	VARIABLE
ATTRIBUTES:	
1	(m) Attribute: Next Element In
2	(m) Attribute: Next Element Out
3	(m) Attribute: Free elements
4	(m) Attribute: Used elements
5	(o) Attribute: ReRead
6	(o) Attribute: ReWrite
SERVICES:	

10.2.4.4.1 Attributes

Variable Objects of this class will have the following constant value of the following (parent class) attribute:

2 VariableTypeIdentifier = FIFO

Octet length

The octet length of a FIFO indicates the length of a single element.

Next Element In

Next Element In is an Integer16, which holds the index number for the next element to be written to the FIFO.

Next Element Out

Next Element Out is an Integer16, which holds the index number for the next element to be read from the FIFO.

Free elements

This attribute is an Integer16 which holds the current number of free elements in the FIFO (= 0 if full).

Used elements

This attribute is an Integer16 which holds the current number of used elements in the FIFO (= 0 if empty).

Reread

Reread is a Boolean. When the value of Reread is TRUE, a Read will return the previously read element, instead of returning and removing the element indexed by Next Element Out. This facility can be used if a transmission error resulted in a faulty read of an element.

Rewrite

Rewrite is a Boolean. When the value of Rewrite is TRUE, a Write will overwrite the previously written element, instead of inserting a new element. This facility can be used if a transmission error resulted in a faulty write of an element.

10.2.5 Services

The services defined in the Variable ASE provide access to attributes and data of Variable Objects. The services are invoked by a REP ASE REQUEST Service invocation. The REP ASE REQUEST Service conveys the Variable ASE Service parameters. Refer to 10.3.3 for the documentation of these parameters.

10.2.5.1 Read

This service is used to read (part of) the specified variable. No data is passed with the request. The value of the specified variable is returned in the response.

10.2.5.2 Write

This service is used to update (part of) the specified variable. The data passed with the request is written into (part of) the specified variable. No data is returned in the response.

10.2.5.3 And

This service performs a logical AND of (part of) the specified variable and the data passed with the request. The result is stored in (part of) the variable. The service is used to clear individual bits in a variable without first reading the variable and afterwards writing the result back to the variable. No data is returned in the response.

10.2.5.4 Or

This service performs a logical OR of (part of) the specified variable and the data passed with the request. The result is stored in (part of) the variable. The service is used to set individual bits in a variable without first reading the variable and afterwards writing the result back to the variable. No data is returned in the response.

10.2.5.5 Test-And-Set

This service performs a logical AND of (part of) the specified variable and the data passed in the request. The result of the AND operation is returned in the response. Then a logical OR operation of (part of) the specified variable and the data passed in the request is performed. The result of the OR operation is stored in (part of) the variable. The service is used to set individual bits in a variable, and in the same transmission read their state prior to the operation.

10.2.5.6 Get Variable Attribute

This service is used to read an attribute of the specified Variable Object. The attribute is specified by the offset parameter.

10.2.5.7 Set Variable Attribute

This service is used to update an attribute of the specified Variable Object. The attribute is specified by the offset parameter.

10.3 Type 4 Route Endpoint ASE

10.3.1 Overview

In the Fieldbus environment, application processes contain data that remote applications are able to read and write. The data is accessed by means of the REP ASE. This ASE offers services to user applications in clients. The services REQUEST and RESPONSE give the access to real Variable Objects (VAOs) located in a server. In the server the REP ASE conveys data to / from the real Variable Object and returns responses. The services Get REP Attribute, Set REP Attribute, Reserve REP and Free REP are all for local use.

10.3.2 Route Endpoint Model

10.3.2.1 Description

The Route Endpoint class is defined to support the on-demand exchange of confirmed and unconfirmed services between two application processes. It uses AR ASE services for conveying APDUs.

An REP is an object container, containing Variable Objects of a variable type. It performs the access to variable objects. The Variable Object ID is unique within an REP. The request service addresses a Variable Object within the REP. The Variable Object ID is one of the request service parameters.

An REP in the proxy container role represents a remote REP in a Server. The remote REP is in the real container role. The Destination Route attribute of the REP in the client holds the Route to the real container REP, which it represents.

The REP ASE handles the segmentation when the data length exceeds the MaxDataSize.

When data is segmented, the order of segments is indicated by the offset part of the APDU.

To initiate access of a proxy Variable Object, the user application uses Reserve REP to get a REP currently not in use.

The attributes of the REP are set, including the Destination Route to the remote REP. This is done by the Set REP Attribute service. The requesting user then issues a REQUEST primitive with the REP ID and a Variable Object ID as parameters. The REP ASE builds a request APDU, and passes that along with Route Info in an AR Send request.

Following a successful transmission, if the REQUEST was confirmed, the AREP receives a response APDU and conveys that to the requesting REP by an AR Send indication. The user application can now read the result by issuing a RESPONSE primitive, with the REP ID as a parameter.

10.3.2.2 Route Endpoint Formal Model**FAL ASE:** **Route Endpoint ASE****CLASS:** Route Endpoint**CLASS ID:** 1**PARENT CLASS:** TOP**ATTRIBUTES:**

1. (m) Key Attribute: Endpoint Address
2. (m) Attribute: Role (Proxy object / Real object)
3. (m) Attribute: REP State
4. (m) Attribute: Priority
5. (m) Attribute: Confirmation
6. (m) Attribute: Destination Route
7. (m) Attribute: Source route
8. (m) Attribute: Progress
9. (m) Attribute: Capabilities
10. (m) Attribute: Flat addressing
11. (o) Attribute: Remote LAN Server ID

SERVICES:

1. (m) OpsService: REQUEST
2. (m) OpsService: RESPONSE
3. (m) OpsService: Reserve REP
4. (m) OpsService: Free REP
5. (m) OpsService: Get REP Attribute
6. (m) OpsService: Set REP Attribute

10.3.2.2.1 Attributes**Endpoint Address**

This key attribute holds the Endpoint address identifying the REP.

ROLE

This attribute specifies the role of the REP. The valid values are:

Proxy container	Endpoints of this type are used for sending requests to servers and receive responses from them. The objects contained in REPs of this role are proxy objects for the real objects in the server.
Real container	Endpoints of this type are used for receiving confirmed and unconfirmed requests from clients and sending responses to them. The objects contained in REPs of this role are the real Variable Objects.

REP State

This attribute indicates the state of the REP. The values for this attribute are: IDLE RESERVED, WAITING FOR RESPONSE, RESPONSE RECEIVED or NOT IN USE.

Priority

The DLL may provide the possibility to send high priority APDUs before APDUs of lower priority. The priority only refers to the request on the local link, not requests passing gateways and not the response.

Confirmation

This attribute indicates whether the request has to be confirmed or unconfirmed. The response is always returned unconfirmed. If the Destination Route contains one or more broadcast addresses (126) this attribute must be set to unconfirmed.

Destination Route

The Destination Route describes the Route to the destination REP. It is a sequence of Endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the next DLE, AREP or REP to receive the APDU. In a request, the Destination Route holds the Route to the REP to respond. In a response, Destination Route holds the Route to the requesting REP.

Source Route

The Source Route describes the Route to the source REP. It is a sequence of endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the DLE, AREP or REP endpoint from where the APDU was conveyed. In a request, the Source Route holds the Route to the requesting REP. In a response, Source Route holds the Route to the responding REP.

Progress

This attribute is only relevant in proxy container REPs. It indicates the progress of a request. It indicates number of segments that has been delivered divided with the total number of segments. This means for non-segmented requests the value is zero while waiting for the response, and one when the response is received. For segmented requests, the value will start from zero and gradually increase till it reaches the value of one.

Capabilities

This attribute defines the capabilities of the REP addressed by Destination Route. It is a local attribute, which must be set up by the user application, to reflect the capabilities of the REP containing the real Variable Objects. The value of Capabilities is used by the requesting REP to build the APDU. The attribute indicates, whether the responding REP is capable of handling bit addressing or not.

Flat addressing

This attribute indicates, whether the REP should be seen as a container of individual Variable Objects, or as a container of one, flat memory area. If Flat addressing is selected, the Variable Object ID parameter of the REQUEST service indicates the offset in octets from the beginning of the memory area.

Remote LAN Server ID

This optional attribute specifies the ID of a LAN server. This attribute is only used when the Client is directly connected to a LAN network and uses this LAN network as a transport medium to a fieldbus device/gateway. The ID may be an Internet URL address or a numerical IP/IPX address.

10.3.3 Route Endpoint ASE Service Specification**10.3.3.1 REQUEST Service**

This service is used to initiate access of the value of a Variable Object. It may be used for as well reading as updating the value. The result is the operation (if any) is retrieved by the RESPONSE service.

10.3.3.1.1 Service Primitives

The service parameters for this service are shown in Table 290. The Indication parameters are indicating the parameters in the APDU received by the remote REP. These parameters are for local use within the remote REP and its variable objects.

Table 290 – REQUEST Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M		
REP	M	M		
Variable object ID	M	M(=)		
Variable Service	M	M(=)		
Data length	M	M(=)		
Offset/Attribute	C	C(=)		
Bit-no	C	C(=)		
Data	C	C(=)		
Result				
Status				M

Argument

The argument carries the parameters of the REQUEST service invocation.

REP

This parameter is the EP address of the REP holding the route information.

Variable ID

This parameter identifies the Variable Object. The Variable ID indicates the Variable Object on which the service is to be performed.

Variable Service

This parameter holds the Type 4 Variable service to be performed. (Read, Write, And, Or, Test-And-Set, Get attribute and Set Attribute).

Data Length

This parameter indicates the number of octets of data to be transferred.

Offset/Attribute

This parameter is used when accessing only part of a structured variable or an Attribute of the variable. When accessing part of a structured variable, it indicates the offset in octets to the starting octet of the data block to be transferred, relative to the first octet of the variable. When accessing attributes, it identifies the attribute to be accessed.

Bit-no

This parameter is used to select a bit within a BitString. It indicates the bit number (1-8) within an octet. The Offset/Attribute parameter is used to select the octet.

Data

This parameter holds the data to be transferred.

Status

As a result of the request, Status is returned indicating OK or FAILURE.

10.3.3.2 RESPONSE Service

This service is used to return the result of a confirmed REQUEST to the requesting user application in a requester.

10.3.3.2.1 Service Primitives

The service parameters for this service are shown in Table 291. The response parameters are for local use within the responding REP and its variable object.

Table 291 – RESPONSE Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument			M	M
REP			M	M
Result				
Variable Service			M	M
Data length			M	M(=)
Error status			M	M(=)
Data			C	C(=)

Argument

The argument carries the parameters of the service request.

REP

This parameter holds the ID of the REP holding the route information.

Variable Service

This parameter holds the Type 4 Variable service that has been performed. (Read, Write, And, Or, Test and Set, Get attribute and Set Attribute).

Data Length

This parameter indicates the number of octets of data that have been transferred.

Error status

This parameter holds error information for errors occurred in any layer local or remote. For details, see Table 292.

The error code can be generated anywhere on the route between client and server, or by the responding Variable Object.

Table 292 – Error Codes by source

Error description	Client	Gateway	Server
User Application:			
Instruction Error	X		X
Info length error	X		X
FIFO full or empty			X
Write protection			X
Data format error			X
Variable Object ID error			X
Time Out	X		
Actual data error			X
Historical data error			X
FAL:			
Route error	X	X	X
DL-layer:			
No Response	X	X	
WAIT TOO LONG	X	X	
Out of sync	X	X	
CRC ERROR	X	X	
DLE not Client	X	X	
Physical Layer:			
NET SHORTCIRCUIT	X	X	
OVERRUN/FRAMING ERROR	X	X	
RS-232 HANDSHAKE ERROR	X	X	X

Data

This parameter holds the data that has been transferred.

10.3.3.3 Reserve REP Service

This service is used by the user application to reserve an REP currently not in use, and to get the ID of the reserved REP.

10.3.3.3.1 Service Primitives

The service parameters for this service are shown in Table 293.

Table 293 – Reserve REP Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Result				
REP				M

REP

This parameter holds the ID of the reserved REP. If no REP is available, zero is returned.

10.3.3.4 Free REP Service

This service is used by the user application to free an REP by changing the state of the REP to “not in use”.

10.3.3.4.1 Service Primitives

The service parameters for this service are shown in Table 294.

Table 294 – Free AREP Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M			
REP		M		

Argument

The argument carries the parameters of the service request.

REP

This parameter holds the EP address of the reserved REP to free.

10.3.3.5 Get REP Attribute Service

This service is used by the user application to read an attribute of a REP locally.

10.3.3.5.1 Service Primitives

The service parameters for this service are shown in Table 295.

Table 295 – Get REP attribute Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M			
REP	M			
Attribute index	M			
Result (+)				
Value				C
Result (-)				
Status				C

Argument

The argument carries the parameters of the service request.

REP

This parameter specifies the REP.

Attribute index

This parameter identifies the attribute to read.

Value

This parameter returns the value of the specified attribute.

Status

This parameter returns error information if the service fails. The possible values are: Illegal attribute index, Illegal REP address.

10.3.3.6 Set REP Attribute Service

This service is used by the user application to write a value into an attribute of a REP locally.

10.3.3.6.1 Service Primitives

The service parameters for this service are shown in Table 296.

Table 296 – Set REP attribute Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M			
REP	M			
Attribute index	M			
Result				
Status				M

Argument

The argument carries the parameters of the service request.

REP

This parameter specifies the REP.

Attribute index

This parameter identifies the attribute to update within the specified REP.

Value

This parameter holds the value to be written to the specified attribute.

Status

As a result of the request, Status is returned indicating OK or the reason for failure. The possible values are: OK, Illegal attribute index, Illegal value, Illegal REP address.

10.4 Type 4 Application Relationship ASE**10.4.1 Overview**

To support access to the remote AP, the Application Relationship ASE is defined. It provides services to the AP for accessing communication-related parameters, and it provides services to the REP ASE for conveying service requests and responses.

The AR ASE provides services at the endpoints of ARs (AREPs).

10.4.2 Application Relationship Class Specification

10.4.2.1 Application Relationship Formal Model

The functionality of the AR ASE is described in 10.1.

The application ASE defines one class, the AREP class.

FAL ASE: **Application Relationship ASE**

CLASS: **AREP**

CLASS ID: 1

PARENT CLASS: TOP

ATTRIBUTES:

1. (m) Key Attribute: Endpoint Address
2. (m) Attribute: Role (Client, Server, Peer)
3. (m) Attribute: DLL Reference
4. (m) Attribute: MaxPDUSize
5. (m) Attribute: MaxDataSize
6. (m) Attribute: Acknowledgement
7. (m) Attribute: MaxIndicationDelay
8. (m) Attribute: Local DLE address (P-NET/IP/IPX address)
9. (o) Attribute: MaxRetryTime
10. (o) Attribute: MaxRetries
11. (o) Attribute: MaxOutstandingRequests
12. (o) Attribute: BaudRate
13. (o) Attribute: NumberOfClientDLEs

SERVICES:

- 1 (m) OpsService: AR-Get Attribute
- 2 (m) OpsService: AR-Set Attribute
- 3 (m) OpsService: AR-Send
- 4 (m) OpsService: AR-Acknowledge

Endpoint Address

This attribute specifies the numeric identifier of the AREP. It is used by the FAL to select the AREP, and implicitly the DLE for a request.

Role

This attribute specifies the role of the AREP. The valid values are:

Client Endpoints of this type send confirmed and unconfirmed Request APDUs to servers and receive Response APDUs.

Server Endpoints of this type receive confirmed and unconfirmed Request APDUs from clients and send unconfirmed Response APDUs.

Peer Endpoints of this type act as both Clients and Servers.

MaxPDUSize

This attribute specifies the maximum PDU size that can be sent by the DLL.

MaxDataSize

This attribute specifies the maximum data size that can be sent by the DLL. If the data length exceeds MaxDataSize, the VARIABLE ASE shall segment the request.

DLL Reference

This attribute contains the necessary context to convey the DLL relations.

Acknowledgement

This attribute describes the type of Acknowledgement to be used by DLL: Acknowledged or Unacknowledged transfer of Confirmed APDUs, and Acknowledged or Unacknowledged transfer of Unconfirmed APDUs.

MaxIndicationDelay

This attribute indicates to the user application, how long time a Variable Object can use to prepare a response after receiving an indication requiring that. If the Variable Object is unable to prepare a response within MaxIndicationDelay, it must issue an AR-Acknowledge. The value of MaxIndicationDelay is calculated by the DLE.

Local DLE address (P-NET/IP/IPX address)

This attribute reflects the DL address of the related DLE. The format varies depending on if it is a P-NET address or a LAN address. In some situations it may be a read-only attribute.

MaxRetryTime

This attribute indicates the maximum time the DLE should try to re-transmit a request as a result of Acknowledge responses from the remote Variable Object.

MaxRetries

This attribute indicates the maximum number of re-transmissions carried out by the DLE as a result of transmission errors.

MaxOutstandingRequests

This attribute indicates the maximum number of requests that the AREP may initiate without receiving the related responses.

BaudRate

This attribute specifies the baud rate used by the physical layer. It is conveyed to/from the physical layers by the DLEs.

NumberOfClientDLEs

This attribute only relates to a DLE using the P-NET protocol in the client role. It specifies the number of participants in a token round.

10.4.3 Application Relationship ASE Service Specifications

This section contains the definition of the services that are unique to this ASE. The services are:

AR Send

AR Acknowledge

AR Get Attribute

AR Set Attribute

10.4.3.1 Common parameter definition

Parameters used in more AR ASE services are defined below:

Route info

The complete DL Route contains the following elements:

Destination Route

The Destination-Route describes how to reach the destination REP. It is a sequence of Endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the next DLE, AREP or REP to receive the APDU.

Source Route

The Source Route in the same way describes how to find the way back to the endpoint that initiated the request. It is a sequence of endpoint addresses and DL-addresses. On its way to the destination, the first element always indicates the address of the DLE, AREP or REP endpoint from where the APDU was sent.

Priority

The underlying layer may provide the possibility to send high priority APDUs before APDUs of lower priority. This priority only refers to the request on the local link, not requests passing gateways and not the response.

Confirmation

This parameter indicates whether the request has to be confirmed or unconfirmed. A response is always returned unconfirmed. If the destination route contains one or more broadcast addresses (126) this attribute must be set to unconfirmed.

Remote LAN Server ID

This optional parameter specifies the ID of a LAN server. This attribute is only used when the Client is directly connected to a LAN network and uses this LAN network as transport medium to a Fieldbus device/gateway. The ID may be an Internet URL address or numerical IP/IPX address.

APDU Header

The APDU Header parameter holds a Control/Status subfield indicating the Variable Service, an APDU format subfield, indicating whether the APDU holds an offset, and finally an APDU length subfield, indicating the octet length of the APDU.

APDU Body

This parameter holds the APDU Body to send or receive.

10.4.3.2 AR Send Service

The AR Send service is used to send confirmed or unconfirmed APDUs from one AREP to another.

It is the task of the receiving AREP to convey the APDU to the Endpoint specified by the first element in the destination route. In the gateway situation the AR is using this AR Send service to convey the APDU to an AREP.

10.4.3.2.1 Service Primitives

The service parameters for this service are shown in Table 297.

Table 297 – AR Send Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M	M		
Route info	M	M		
APDU Header	M	M(=)		
APDU Body	C	C(=)		
Result				
Status				M

Argument

The argument carries the parameters of the service request.

Status

As a result of the request, Status is returned indicating OK or the locally detected reason for failure. Status only refers to local conditions. The possible values are: OK, Route Error.

10.4.3.3 AR Acknowledge Service

When the Variable ASE in a confirmed request is not able to process the indication and respond within the time limit, "MaxIndicationDelay", it must submit an AR Send Acknowledge request primitive to the local AREP, and in this way free the Token. "MaxIndicationDelay" is an attribute of the AREP.

10.4.3.3.1 Service Primitives

The service parameters for this service are shown in Table 298.

Table 298 – AR Acknowledge Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M			
Route info	M			

Argument

The argument carries the parameters of the service request.

10.4.3.4 AR-Get Attribute Service

This confirmed service is used to read the value of attributes of an AREP locally.

10.4.3.4.1 Service Parameters

The service parameters for the AR Get Attribute Service are shown in Table 299.

Table 299 – AR Get Attributes Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Attribute index	M			
Result (+)				S
Value				C
Result (-)				S
Status				C

Argument

The argument carries the parameters of the service request.

AREP

This parameter identifies the AREP by its endpoint address.

Attribute index

This parameter identifies the attribute.

Result (+)

This selection type parameter indicates that the request succeeded.

Value

This parameter is returned with the requested attribute value if the request succeeded.

Result (-)

This selection type parameter indicates that the request failed.

Status

If the request failed, Status is returned indicating the reason for failure. The possible values are: Illegal attribute index, Illegal AREP address.

10.4.3.5 AR Set Attributes Service

This confirmed service is used to set the current value of attributes of an AREP locally.

10.4.3.5.1 Service Parameters

The service parameters for the AR Set Attribute Service are shown in Table 300.

Table 300 – AR Set Attributes Service Parameters

Parameter Name	Req	Ind	Rsp	Cnf
Argument	M			
AREP	M			
Attribute index	M			
Value	M			
Result				
Status				M

Argument

The argument carries the parameters of the service request.

AREP

This parameter identifies the AREP by its endpoint address. The legal values are 1 to 16 inclusive.

Attribute index

This parameter identifies the attribute to be updated.

Value

This parameter holds the new attribute value.

Status

As a result of the request, Status is returned indicating OK or the reason for failure. The possible values are: OK, Illegal attribute index, Illegal value, Illegal AREP address.

11 Communication Model Type 5 Specification

11.1 Type 5 Concepts

This section introduces the concept of the Type 5 FAL application entity (FAL AE). The Type 5 FAL AE is composed of Type 5 specific APO ASEs and ARs and a subset of the Type 1 APO ASEs. A subset of the data types defined in Clause 4 are supported. The Type 1 ASEs, classes, and services supported by Type 5 are defined in clauses 11.4 and 11.5 below.

The Type 5 FAL AE maps the Type 5 and Type 1 ASEs onto a set of underlying connection-oriented and connectionless communication services modelled as sockets. These services are referred to as socket services in the remainder of this clause. Socket services are independent of the actual protocols used to provide their services. Any protocols that can be mapped to these services, including TCP/UDP, can be used.

Type 5 supports application access to both Type 1 and Type 5 field devices. Access to Type 1 field devices is provided through a *Linking Device*. Linking Devices contain a Type 5 FAL AE that provides a gateway function between Type 5 and Type 1 networks.

11.1.1 Objectives of the Type 5 FAL AE

The primary objective of the Type 5 FAL AE is to convey Type 5 and Type 1 ASE services using socket services. This allows Type 5 and Type 1 field devices, conventional I/O devices, and other I/O devices to be connected to the Type 5 through a Type 5 FAL AE in a *Linking* or *Gateway Device*.

Another objective of the Type 5 FAL AE is to republish Type 1 data from one Type 1 link onto another in multi-ported Type 1 devices that do not support Type 1 data link layer bridging. This allows a Type 5 FAL AE in a Linking Device to integrate multiple standalone Type 1 interfaces instead of using a Type 1 bridge.

Another objective of the Type 5 FAL AE is to send and receive messages to support applications responsible for adding and deleting Type 5 application objects to/from the network, and for locating application objects on the network.

Another objective of the Type 5 FAL AE is to send and receive diagnostic messages to support LAN redundancy applications.

As a result, the Type 5 FAL AE enables:

- control systems to be constructed from Type 5 and Type 1 devices
- Type 5 and Type 1 networks to be linked by Linking Devices and
- remote applications can access field devices of any type through socket services using the Type 5 Communications Model.

11.1.2 Type 5 Devices

Type 5 devices are FAL devices that contain a Type 5 FAL AE.

One class of Type 5 devices is capable of interconnecting a Type 5 network to one or more Type 1 links. These Type 5 devices are called Linking Devices. The Type 5 FAL AE in Linking Devices provides gateway services that map Type 5 APDUs to their Type 1 counterparts.

Another class of Type 5 devices is capable of interconnecting a Type 5 network to links that support other data link layer types. These Type 5 devices are called Gateway Devices. The Type 5 FAL AE in Gateway Devices provides gateway services that map Type 5 APDUs to their counterparts on the other links.

Another class of Type 5 devices is capable of supporting redundant LAN connections. These Type 5 devices are called LAN Redundancy Capable Devices. The Type 5 FAL AE in these devices provides LAN Redundancy ASE services to the LAN Redundancy entity in the device.

11.1.3 Data Type ASE

11.1.3.1 Data Types

The data types supported by Type 5 are a subset of those defined in clause 4. They are:

- Boolean
- Integer8
- Integer16
- SM Time
- Unsigned8
- Unsigned16
- Unsigned32
- VisibleString
- OctetString

11.1.3.2 Nesting Level

One level of nesting is supported.

11.1.4 APO ASEs

The Type 5 FAL AE supports Type 5 and Type 1 APO ASEs as specified in clauses 11.4 and 11.5 below. The Type 5 ASEs are the S-VFD, SMK and LAN Redundancy ASEs. The Type 5 S-VFD ASE is an adaptation of the Type 1 AP ASE. The Type 5 FAL AE also supports versions of the QUB-CO and QUU Type 1 ARs that are adapted for use over sockets.

11.1.4.1 Type 5 LAN Redundancy ASE

The Type 5 Redundancy ASE specifies services that support LAN Redundancy entities. A LAN Redundancy Entity in LAN redundancy capable devices uses the services of the Type 5 FAL AE to send and receive LAN redundancy ASE APDUs. These APDUs are used by the LAN Redundancy Entity to construct a network status table used to choose routes that avoid network faults.

11.1.4.2 System Management Kernel (SMK) ASE

11.1.4.2.1 SMK

The SMK entity is the entity in a device that is responsible for system management operation in a device. The SMK ASE defines a set of services that support the conveyance of system management requests and responses to and from SMK entities.

When resident in a Type 5 device, the SMK services are conveyed by Type 5 ARs. When resident in a device of another type, the SMK services may be conveyed by direct access to the data link layer connectionless services.

Like Type 1 ASE services, SMK ASE services can be conveyed between a device connected to a Type 5 network and a device connected to a network that uses an underlying Type 1 Data Link Layer. In this case, the Type 5 FAL AE in a Linking Device receives an SMK service request or response from one network and maps it for conveyance by the other.

11.1.4.2.2 SMK ASE Services

One SMK ASE service is defined for identifying devices.

Two SMK ASE services are defined for adding and deleting Type 5 Devices and Type 1 devices connected to a Linking Device.

- One of these services is used by the SMK in a new device to announce its presence periodically on the network.
- The other service is used by a configuration application receiving the annunciation to return basic configuration information to the SMK. This allows the SMK to learn its role in the system and begin normal operations.

Two more SMK ASE services are defined for finding devices and named objects on the network.

11.1.4.3 Simple Virtual Field Device (S-VFD) ASE

11.1.4.3.1 S-VFD

The Type 5 S-VFD is a Type 5 AP. It is a simple form of the AP object defined by the Type 1 AP ASE. The S-VFD provides a subset of the services defined by the other Type 1 ASEs. These services are conveyed on Type 5 networks using the Type 5 syntax and procedures defined in IEC 61158-6.

11.1.4.3.2 S-VFD Contexts

Type 5 S-VFD Contexts are the Type 5 version of the Type 1 AP Contexts (defined by the Type 1 AP ASE). Type 5 S-VFD Contexts provide a Type 1 FAL service interface for access to both Type 5 S-VFDs and Type 1 APs. Type 1 APs are accessed through Type 5 FAL AEs in Linking Devices. Three types of S-VFD Contexts are defined, Client/Server, Publisher/Subscriber, and Report Distribution.

11.1.4.3.2.1 Client / Server S-VFD Contexts

The Type 5 FAL AE supports both dynamically established and preconfigured Client / Server S-VFD Contexts, as follows. Clients send requests to initiate S-VFD Contexts to the server FAL AE. If the requested server S-VFD Context has been preconfigured, the FAL AE delivers the request to it for processing. If not, the FAL AE dynamically creates a new server S-VFD Context and delivers the request to it for processing.

11.1.4.3.2.2 Publisher / Subscriber S-VFD Contexts

Each Type 5 subscriber S-VFD Context endpoint is configured to receive from a specific subscriber Type 5 AR. Each subscriber Type 5 S-VFD Context endpoint, like its Type 1 counterpart, is configured to receive its messages from a single Type 5 publisher S-VFD Context endpoint. The original publisher may exist on a Type 1 link when a Linking Device is used to republish the data onto the Type 5 network. In addition, if the Type 5 subscriber is located in a Linking Device, it may be configured to republish the data onto a Type 1 link.

When conveyed through a Linking Device, the FAL AE maps Type 1 publisher/subscriber AP Contexts/ARs to Type 5 publisher/subscriber S-VFD Contexts, receiving published data on one and republishing it on the other.

The Type 5 FAL AE can be configured for the three types of publishing below.

- A local Type 5 application can publish data onto the Type 5 network through the Type 5 FAL AE.
- The Type 5 FAL AE in a Linking Device can republish data from a Type 1 device onto the Type 5 network.
- The Type 5 FAL AE in a Linking Device can republish data from a Type 5 device onto a Type 1 link.

11.1.4.3.2.3 Report Distribution S-VFD Contexts

Report source and sink Type 5 S-VFD Contexts operate analogously to publisher/subscriber Type 5 S-VFD Contexts.

The Type 5 FAL AE can be configured for the three types of report distribution below.

- A local Type 5 application can send reports onto the Type 5 network through the Type 5 FAL AE.
- The Type 5 FAL AE in a Linking Device forward reports received from devices on a Type 1 link onto the Type 5 network.
- The Type 5 FAL AE in a Linking Device can forward reports received from a Type 5 device onto a Type 1 link.

11.1.4.3.3 S-VFD ASE Services

The S-VFD ASE provides services to identify and obtain the status of S-VFDs. These services are the same as those provided by the Type 1 AR ASE. The S-VFD ASE also supports an Initiate service to open an S-VFD Context that is similar to the Initiate service provided by the Type 1 AP ASE.

The Type 5 Initiate service can be directed to a generic address that allows multiple client applications operating over independent Type 5 ARs to open concurrent access to the same S-VFD. The S-VFD ASE operating at that address creates and a new S-VFD Context to process each received initiate request.

The Type 5 Initiate service operates locally for publisher/subscriber S-VFD Contexts. It opens the context for publishing or subscribing.

11.1.5 Application Relationships

Type 5 AP Context APDUs are conveyed using Type 5 ARs that operate over the underlying socket services.

A Type 5 AR provides a communications path to one or more Type 5 devices. Type 5 ARs support Type 5 S-VFD Contexts in the same way that the Type 1 ARs support Type 1 AP Contexts, with the exception that a single Type 5 AR can support more that one S-VFD Context at a time. Figure 71 illustrates this concept.

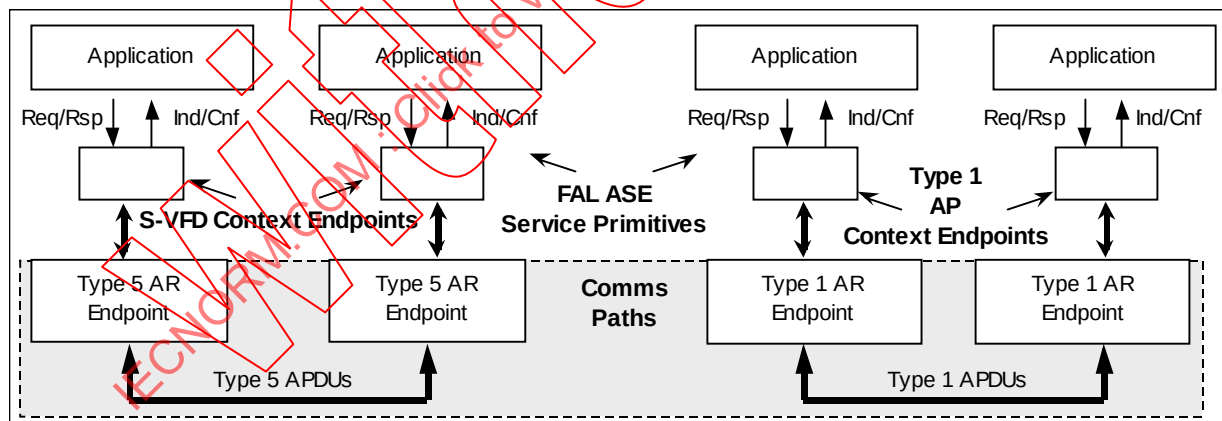


Figure 71 – Type 5 ARs Provide Communication Path for Type 5 S-VFD Contexts

Two types of Type 5 ARs are defined, client / server and publisher / subscriber. Client / server Type 5 ARs support client / server S-VFD Contexts. Publisher/subscriber Type 5 ARs support both publisher / subscriber and report distribution S-VFD Contexts. Separate Type 5 ARs are not defined for report distribution.

11.1.5.1 Client/Server Application Relationships

Client applications use the Establish AR service to explicitly open Type 5 ARs with remote Type 5 FAL AEs. Client Type 5 AR endpoints are the initiators of the Type 5 AR. Once opened, applications at both endpoints may initiate client/server S-VFD Contexts.

Once the Type 5 AR has been opened, the application can submit the Type 5 S-VFD ASE Initiate service to open access to one or more S-VFDs. On a single client / server Type 5 AR, the client application may have more than one S-VFD context open concurrently.

The requesting application also can direct the Initiate Request to a Type 1 AP if there is an intervening Linking Device. In this case, the intervening Type 5 FAL AE in the Linking Device maps the Type 5 Initiate Request to the appropriate Type 1 AP Context/AREP. Once the context is open, Type 1 ASE services may be sent to the remote Type 5 S-VFD or to the Type 1 AP (through the Type 5 FAL AE in a Linking Device).

Configuration Type 5 ARs are client/server Type 5 ARs used to update the System Management Information Base (MIB) for a device. The Type 5 FAL AE allows only one application to have a configuration Type 5 AR open at a time. However, Type 5 ARs that convey SMK ASE services to read the MIB are independent from configuration Type 5 ARs and are permitted to operate in parallel with configuration Type 5 ARs.

11.1.5.2 Publisher/Subscriber Application Relationships

Type 5 AR endpoints that support publishing/subscribing and sending/receiving reports are configured instead of being dynamically established. They typically multicast their messages, but they can be configured to operate between a sender/receiver pair in a connection-oriented fashion.

Each publisher Type 5 AR endpoint is configured to support one or more publisher or report source S-VFD Contexts. Likewise, each subscriber Type 5 AR endpoint is configured to support one or more subscriber or report sink S-VFD Contexts.

11.1.6 Support for Configuration Management

Configuration data for the Type 5 FAL AE is maintained in the MIB. The Type 5 FAL AE also uses the bridge data structures in the Type 5 MIB if it is in a Linking Device and is not supported by Type 1 bridging capabilities. It uses the bridge data structures to perform republishing and report forwarding operations normally performed by the bridge.

11.2 Type 5 ASEs

11.2.1 Simple Virtual Field Device ASE

11.2.1.1 Overview

This standard models a Simple Virtual Field Device (S-VFD) as a simplified Type 1 Fieldbus Application Process. The S-VFD contains a simplified set of information and services defined for the more general Type 1 AP.

The Application Service Element in the FAL that provides these services is called the S-VFD ASE. In the S-VFD ASE, the S-VFD is modelled and accessed as an APO with a standardised and predefined identifier.

11.2.1.2 Simple Virtual Field Device Model Class Specifications

This subclause provides formal class definitions for Simple Virtual Field Device class and the S-VFD Context class. S-VFD Contexts provide network access to the Simple Virtual Field Devices. They are analogous to Type 1 AP Contexts. Each class definition specifies the attributes and the FAL services defined for it.

11.2.1.2.1 S-VFD Formal Model

The S-VFD class specifies the attributes and services similar to, and sometimes identical to those defined for the AP ASE. Its parent class "top" indicates the top of the FAL class tree. The numeric identifier for the S-VFD Object is always 0 because it is an application process. The use of this reserved value permits management services to be used for all Fieldbus Application Processes without knowing their names.

FAL ASE: **S-VFD ASE**
CLASS: **S-VFD**
CLASS ID: Not used
PARENT CLASS: TOP

IDENTIFY, GET STATUS, and STATUS NOTIFICATION SERVICE ATTRIBUTES:

- 1 (m) Attribute: List Of AP Service Attributes
 - 1.1 (m) Attribute: Manufacturer Identifier
 - 1.1.1 (m) Attribute: Vendor Name
 - 1.1.2 (m) Attribute: Model Identifier
 - 1.1.3 (m) Attribute: Vendor Revision
 - 1.1.4 (o) Attribute: Serial Number
 - 1.2 (o) Attribute: AP Descriptor Reference
 - 1.3 (m) Attribute: Network Access State
 - 1.3.1 (m) Attribute: Service Access Status
 - 1.3.2 (m) Attribute: Operational Status

SYSTEM MANAGEMENT ATTRIBUTES:

- 2 (m) Attribute: List Of System Management Attributes
 - 2.1 (m) Attribute: List Of Configured S-VFD Context Endpoints
 - 2.2 (m) Attribute: List Of Dynamically Established S-VFD Context Endpoints
 - 2.3 (m) Attribute: List Of Configured AR Endpoints
 - 2.4 (m) Attribute: List Of Dynamically Established AR Endpoints

OBJECT MANAGEMENT ATTRIBUTES:

- 3 (m) Attribute: List Of Managed AP Attributes

SERVICES:

- 1 (o) Ops Service: Identify
- 2 (o) Ops Service: Get Status
- 3 (o) Ops Service: Status Notification

11.2.1.2.1.1 Identify, Get Status, and Status Notification Service Attributes

These attributes are defined by the Type 1 AP Class.

11.2.1.2.1.2 System Management Attributes

List Of System Management Attributes

The following attributes are accessible through System Management. They are used by the S-VFD to manage its S-VFD Contexts and its ARs. They are not otherwise accessible.

List of Configured S-VFD Context Endpoints

This attribute contains a list of configured S-VFD Contexts associated with the AP. See the S-VFD Context class below for its attributes.

List of Dynamically Established S-VFD Context Endpoints

This attribute contains a list of dynamically established S-VFD Context associated with the AP. See the S-VFD Context class below for its attributes.

List of Configured AR Endpoints

This attribute contains a list of configured AREPs associated with the AP. See the Type 5 AREP class below for its attributes.

List of Dynamically Established AR Endpoint Info

This attribute contains a list of dynamically established AREPs associated with the AP. See the Type 5 AREP class below for its attributes.

11.2.1.2.1.3 Object Management Attributes

List Of Managed AP Attributes

These attributes are identical to those of the same name defined for the Type 1 AP Class.

11.2.1.2.1.4 Services

The S-VFD services are those defined by the Type 1 AP Class.

11.2.1.2.2 S-VFD Context Formal Model

The S-VFD Context class specifies the attributes and services of the AP Context (as defined for the Type 1 AP) for the S-VFD. Its parent class "top" indicates the top of the FAL class tree.

FAL ASE:

S-VFD ASE

CLASS:

S-VFD Context

CLASS ID:

Not used

PARENT CLASS:

TOP

SYSTEM MANAGEMENT ATTRIBUTES:

1	(m)	Attribute:	Local Context Id
2	(m)	Attribute:	Related AREP Id
3	(m)	Attribute:	Context Type
4	(m)	Attribute:	Context State
5	(m)	Attribute:	Conveyed Reference
6	(m)	Attribute:	Conveyed Reference Subnet Id
7	(m)	Attribute:	User Id
8	(c)	Constraint	Context Type = AP SERVER TYPE 1 CLIENT
8.1	(m)	Attribute:	Generic (TRUE, FALSE)
9	(c)	Constraint	Context Type = AP PUBLISHER TYPE 1 SUBSCRIBER
9.1	(m)	Attribute:	On Change Threshold

SERVICES:

1	(o)	Ops Service:	Initiate

11.2.1.2.2.1 System Management Attributes

The following attributes are accessible through System Management. They are not otherwise accessible.

Local Context Id

This attribute identifies the local context object.

Related AREP Id

This attribute is the numeric identifier of the related AREPs used to send or receive messages. This value is 0 when the Context Type is used to republish or forward reports between two Type 1 ARs. The value 0 signifies that there is no related Type 5 AR endpoint.

Context Type

This attribute indicates the type of the context. Each of the types uses the Related AREP to convey services. Its values include:

AP CLIENT	The Context receives confirmed request primitives from the AP.
AP SERVER	The Context delivers confirmed indication primitives to the AP for processing.
AP PUBLISHER	The Context publishes data originating in the AP.
AP SUBSCRIBER	The Context subscribes to published data and delivers it to the AP.
AP REPORT SOURCE	The Context distributes reports originating in the AP.
AP REPORT SINK	The Context receives reports and delivers them to the AP.
TYPE 1 SERVER ¹	The Context is a server on a Type 1 link and a client on a Type 5 network. The Context receives confirmed indication primitives from the Type 1 link and forwards them onto the Type 5 network.
TYPE 1 CLIENT ¹	The Context is a client on a Type 1 link and a server on a Type 5 network. The Context receives confirmed indication primitives from the Type 5 network and forwards them onto a Type 1 link.
TYPE 1 SUBSCRIBER ¹	The Context is a subscriber on a Type 1 link and a publisher on a Type 5 network. The Context receives published data from the Type 1 link and republishes it onto the Type 5 network.
TYPE 1 PUBLISHER ¹	The Context is a publisher on a Type 1 link and a subscriber on a Type 5 network. The Context receives published data from the Type 5 network and republishes it onto the Type 1 link.
TYPE 1 REPORT SINK ¹	The Context is a report sink on a Type 1 link and a report source on a Type 5 network. The Context receives reports from the Type 1 link and forwards them onto the 5 network.
TYPE 1 REPORT SOURCE ¹	The Context is a report source on a Type 1 link and a report sink on a Type 5 network. The Context receives reports from the Type 5 network and forwards them onto the Type 1 link.

Notes: 1 TYPE 1 Context Types are present only in Linking Devices

Context State

This attribute specifies the state of the context. Valid states are defined in IEC 61158–6.

Conveyed Reference

This attribute contains a reference that is conveyed with each service to or from the corresponding context endpoint(s). Its values are Context Type dependent. They are:

AP CLIENT	The Id of the associated server S-VFD context or Type 1 AREP.
AP SERVER	Not Used.
AP PUBLISHER	"Published-Data" reference used to identify the published data buffer.
AP SUBSCRIBER	The reference for the publisher data buffer that is "Subscribed-To".
AP REPORT SOURCE	"Reporting S-VFD" reference used to identify the AP sending the report messages.
AP REPORT SINK	Not Used.
TYPE 1 SERVER	The numeric id of the Type 1 Server AREP.
TYPE 1 CLIENT	Not Used.
TYPE 1 SUBSCRIBER	Type 1 data link address of the published data buffer.
TYPE 1 PUBLISHER	Type 5 identifier of the published data buffer.
TYPE 1 REPORT SINK	Type 1 data link address of the AREP sending the reports.
TYPE 1 REPORT SOURCE	This value is dynamic and changes with each report received. It is the Type 5 identifier of the source AP of the last received report.

Conveyed Reference Subnet Id

This attribute identifies the subnet of Conveyed Reference and is used to qualify it for service indication primitives received on the S-VFD Context.

Remote Context Subnet Id

This attribute identifies the subnet of the remote context object.

User Id

This conditional attribute identifies the user of the context. If the Context Type indicates an AP type, then this attribute identifies the local AP that is the user of the context. If the Context Type indicates a Type 1 type, then this the use of this attribute is specified below.

Context Type

TYPE 1 CLIENT
TYPE 1 SERVER
TYPE 1 PUBLISHER
TYPE 1 SUBSCRIBER
TYPE 1 REPORT SOURCE
TYPE 1 REPORT SINK

Id of:

Type 1 Client AREP used to send requests on the Type 1 link.
Type 1 Server AREP used to receive requests from the Type 1 link.
Type 1 Publisher AREP used for publishing onto the Type 1 link.
Type 1 Subscriber AREP used to subscribe to data on the Type 1 link.
Type 1 Report Source AREP used for sending reports onto the Type 1 link.
Type 1 Report Sink AREP used to receive report distributions from the Type 1 link.

Generic

This conditional attribute specifies, when TRUE, that this Context receives Initiate requests from remote clients and creates a new Context to handle it and all subsequent accesses to the S-VFD. When FALSE, the Context is the context that handles the Initiate request and all subsequent accesses to the S-VFD. This attribute is present if the Context Type is AP SERVER or TYPE 1 CLIENT.

On Change Threshold

This conditional attribute is present only in LOCAL PUBLISHER and Type 1 SUBSCRIBER contexts. It is used by the user of the context to determine whether or not to submit published data for transfer. It specifies the percentage of change that must be exceeded before the value of the published data is considered to be changed.

The On Change capability operates as follows. The acceptable difference is calculated by multiplying the published data range, specified by the application, by the On Change Threshold percent. (If the data range was 20 and the On Change Threshold was 10%, the difference would be 2). When a published variable value is available, the absolute value of its difference with the last published values is calculated. (If the new value is 17 and the previous value was 14, then the difference is 3). If this difference is greater than the acceptable difference (3 is greater than 2), then the value is considered to have changed and is submitted for transfer. If the value has not been transferred for at least 1/3 of the Inactivity Close Time period for the Type 5 AR transferring the value, then its latest value is submitted for transfer regardless of the On Change Threshold comparison. Support for the On Change capability is optional, although support for this attribute is mandatory.

11.2.1.2.2.2 Services

Initiate

This optional service is used to open the AP Context for an Application Relationship.

Terminate

This optional service is used to indicate that the AP Context has closed.

11.2.1.3 Simple Virtual Field Device ASE Service Specification

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

Initiate

Terminate

11.2.1.3.1 Initiate Service

This service is used to open context to remote endpoint. The AP can be an S-VFD in a Type 5 device or a Type 1 AP in a Type 1 device that is connected to the Type 5 device by a Type 1 link.

11.2.1.3.1.1 Service Primitives

The service parameters for this service are shown in Table 301.

Table 301 – Initiate

Parameter name	.req	.ind	.rsp	.cnf
Argument	M	M(=)		
Connect Option	M	M(=)		
Conveyed Reference	M	M(=)		
Access Protection Supported Calling	M	M(=)		
Password Calling	M	M(=)		
Access Groups Calling	M	M(=)		
Version Object Definitions Calling	M	M(=)		
AP Descriptor Calling	M	M(=)		
Result(+)			S	S(=)
Context Id			C	C(=)
Version Object Definitions Called			M	M(=)
AP Descriptor Called			M	M(=)
Result(-)			S	S(=)
Context Id			C	C(=)
Error Class and Subclass			M	M(=)
Additional Code			U	U(=)
Additional Description			U	U(=)

Argument

This parameter carries the parameters of the service invocation.

Connect Option

This parameter indicates which option is to be used to open the context. Its values shown below.

Connect Option	Description

Conveyed Reference	This option allows the requesting application to provide the Context Id that is to be used to connect to the AP. If the AP is a Type 5 S-VFD, then the Conveyed Reference is a S-VFD Context Id of either a generic or a specific S-VFD Context. If the AP is a Type 1 AP, then the Conveyed Reference is the Id of the Type 1 Client AREP to be used to access the Type 1 device.

System Management MIB ¹⁾	This option allows the requesting application to indicate that it desires to connect to the System Management MIB. The Conveyed Reference identifies the System Management MIB in the remote Type 5 device or in a Type 1 device attached to it using a Type 1 link.

¹⁾ The System Management MIB can also be accessed using either of the Connect Options by providing the correct information.

Conveyed Reference

This parameter identifies the remote context endpoint. Its value is dependent on the value of the Connect Option

Access Protection Supported Calling

This parameter contains the value of the calling AP's Access Protection Supported attribute if it has one. If it does not, its value is null.

Password Calling

This parameter specifies the password to be used for the access to all objects of the server on this AR. If access with password is not to be used on this AR, its value is null.

Access Groups Calling

This parameter specifies the client's membership in specific access groups. The membership applies for the access to all objects of the server on this AR.

Version Object Definitions Calling

This parameter specifies the version of the client's Object Definitions. Its value is null if the client does not contain Object Definitions.

AP Descriptor Calling

This parameter contains the value of the calling AP's Descriptor Reference attribute if it has one. If it does not, its value is null.

Result(+)

This selection type parameter indicates that the service request succeeded.

Context Id

This conditional attribute identifies the context at the responder that has been established.

Version Object Definitions Called

This parameter shall specify the version of the server's Object Definitions.

AP Descriptor Called

This parameter contains the value of the called AP's Descriptor Reference attribute.

Result(-)

This selection type parameter indicates that the service request failed.

Context Id

This conditional attribute identifies the context at the responder that was requested to be established. It is present when the underlying AREP supports multiple concurrent open contexts.

Error Class and Subclass

This parameter shall provide the reason for the failure.

Additional Code

This optional parameter provides an additional code that is associated with the error.

Additional Description

This optional parameter provides 16 octets of additional information that is associated with the error.

11.2.1.3.1.2 Service Procedure

This service operates through a queue. The Confirmed Service Procedure specified in clause 4 applies to this service. The paragraphs below describe the service specific processing, in the following sequence.

If the Type 5 FAL AE is not able to locate the context identified by the Conveyed Reference, or if the context is not AP SERVER or TYPE 1 CLIENT for, the Type 5 FAL AE returns a negative response.

If a service indication is received to open a context to a System Management MIB on a configuration Type 5 AR and there is already a context established to that MIB that allows updates, then the Type 5 FAL AE refuses the request and returns a negative response.

If the request identifies an AP SERVER, and if the Generic attribute is TRUE, the Type 5 FAL AE constructs a new local AP SERVER and delivers the request to it for processing. If FALSE and the AP SERVER is not already opened or in the process of being opened, the Type 5 FAL AE marks it as opening and delivers the indication primitive from it.

If the request identifies a TYPE 1 CLIENT, the Type 5 FAL AE constructs a new local TYPE 1 CLIENT and associates it with the Type 1 Client AREP. If the Type 1 Client AREP is not already open, the Type 5 FAL AE opens it using the Type 1 Initiate AP ASE service. This permits more than one TYPE 1 CLIENT to be associated with the Type 1 Client AREP.

If the request accepted, the response is returned to the client by the Type 5 FAL AE contains the context id of the newly created local context. This id is to be used by the client for subsequent accesses on the context.

If the request is rejected, the Type 5 FAL AE deletes the context object if it created one, and returns a negative response.

If the context was opened to a System Management MIB on an AR that does not support update requests, the server Type 5 FAL AE issues a negative response if it receives an update request from the client without delivering the service indication to the System Management MIB.

11.2.1.3.2 Terminate Service

This service is the same as that defined in Type 1 for the AP Class, with the exception that the AREP Id parameter is replaced by the Context Id. The Context Id identifies the context to be terminated.

11.2.2 System Management Kernel ASE

11.2.2.1 Overview

The FAL system management kernel model defines an object that supports a system management agent in device. This system management entity is referred to as the System Management Kernel (SMK) because it provides basic capabilities to add or delete the device from the network and to locate objects within the device.

11.2.2.2 FAL System Management Kernel Class Specification

11.2.2.2.1 System Management Kernel Formal Model

FAL ASE: **SYSTEM MANAGEMENT KERNEL ASE**

CLASS: **SYSTEM MANAGEMENT KERNEL**

CLASS ID: Not used

PARENT CLASS: TOP

SYSTEM MANAGEMENT ATTRIBUTES:

1	(m)	Key Attribute:	Device Id
2	(m)	Attribute:	Physical Device Tag
3	(m)	Attribute:	List of Network Addresses
4	(m)	Attribute:	SMK Multicast Address
5	(m)	Attribute:	Annunciation Repeat Time
6	(m)	Attribute:	Device Type
6	(m)	Attribute:	Device Redundancy Capability
7	(m)	Attribute:	Device Redundancy State
8	(m)	Attribute:	Device Index
9	(m)	Attribute:	Max Device Index
10	(m)	Attribute:	SMK Version Number
11	(m)	Attribute:	Device Version Number

SERVICES:

1	(m)	OpsService:	Find Tag Query
2	(m)	OpsService:	Find Tag Reply
3	(m)	OpsService:	Identify
4	(m)	OpsService:	Annunciate
5	(m)	OpsService:	Device Assignment
6	(m)	OpsService:	Clear Address

11.2.2.2.1.1 System Management Attributes

The following attributes are accessible through System Management. They are not otherwise accessible.

Device Id

This key attribute specifies the vendor specific identification for the device.

Physical Device Tag

This attribute specifies the site administered name (tag) assigned to the device.

List of Network Addresses

This attribute specifies a list of network addresses used by the device. More than one address is used when Local Area Network (LAN) redundancy is supported by the device.

SMK Multicast Address

This attribute contains the configured network multicast address used by the SMK.

Annunciation Repeat Time

This attribute specifies the milliseconds between annunciation messages. Its default value is 15000 (15 seconds).

Device Type

This attribute specifies the type of device. The list of valid types includes Type 1 Linking Device, Type 1 Device, Type 5 Device, Gateway Device.

Device Redundancy Capability

This attribute specifies the device's ability to operate as one of a set of redundant devices. All devices in the set are required to have the same value of this attribute configured. Three values are possible, None, External Control, and Automatic.

External Control indicates that the devices in the set are capable of operating as either a primary or a secondary (backup). An external agent is required to synchronise device data, detect a failure of the primary, and after detection, issue a Device Assignment service request to one of the secondaries to change its Device Redundancy State from secondary to primary.

Automatic indicates that the device is capable of synchronising with its redundant partners, detecting a fault in the primary, and automatically selecting a new primary to survive the fault.

Device Redundancy State

This attribute specifies the actual device redundancy state of the device. If the device is not participating in redundancy, its value is None. If it is participating as an external controlled device, then its value is either "external control - primary" or "external control - secondary". If it is participating as an automatic device, then its values are "automatic - primary", "automatic - secondary", or "automatic - member of set". This last value indicates that the devices in the set are to select which device is the primary on their own. The Device Assignment Service can be used to assign the value of this attribute. The assigned value must be compatible with the Device Redundancy Capability attribute.

Device Index

This attribute is a site administered number that identifies the device. This index is unique within an FAL subnet.

Max Device Index

This attribute defines the maximum value that the device index may have. This effectively indicates the maximum number of devices that may be configured into an FAL subnet. This number is a multiple of 32.

SMK Version Number

This attribute is the version number for the attribute values that precede this attribute. Each time the preceding attribute's values change, this version number is incremented by 1. Its value survives a power failure if the attributes it covers survive the power failure.

Device Version Number

This attribute is the version number that represents the composite state of the static information in the applications in the device. Each time there is change to the static data of any application in the device, this version number is incremented by 1. How the SMK determines that a change has occurred is a local matter.

11.2.2.2.1.2 Services

All services defined for this class are mandatory.

Find Tag Query

This unconfirmed service is to search for an object in the network.

Find Tag Reply

This unconfirmed service is used as the reply to the Find Tag Query service. It returns a list of S-VFD Context Identifiers that can be used in the Initiate service to access the queried object.

Identify

This confirmed service is used to request the SMK to identify itself.

Annunciate

This unconfirmed service is used to periodically announce the presence of the SMK on the network.

Device Assignment

This confirmed service is used to send basic configuration information to the SMK.

Clear Address

This unconfirmed service is used to request an SMK to clear its network address.

11.2.2.3 System Management Kernel ASE Service Specification

SMK Services may operate over dynamically established or pre-established client/server Type 5 ARs. They may also operate over publisher/subscriber ARs.

Pre-established ARs are not required to be visible in the SMIB. They may be pre-established, just prior to service invocation, using a set of default attribute values and the destination address of the remote device. Once established, these SMK ARs convey a single service invocation and are then terminated. Each service defines the ARs that it may use.

11.2.2.3.1 Find Tag Query Service

This unconfirmed service is used to send a query to one or more SMKs to locate an object on the network.

This service operates over dynamically established or pre-established client/server Type 5 ARs or over a publisher/subscriber AR. The remote address may be an individual Type 5 device address, the SMK multicast address, or the multicast address used for the Annunciation Service.

11.2.2.3.1.1 Service Parameters

The service parameters for this service are shown in Table 302.

Table 302 – Find Tag Query Service Parameters

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Invoke ID	M	M(=)
Source Address		M
Destination Address	M	M(=)
SMK Id	M	M(=)
Physical Device Tag	S	S(=)
VFD Tag	S	S(=)
S-VFD Reference	S	S(=)
Application Object Tag	S	S(=)
Element ID	C	C(=)

Argument

The argument contains the parameters of the service request.

Source Address

This parameter is the Type 5 address from which the service request was sent. It is used by the responder when returning the reply.

SMK Id

This parameter identifies which SMK are to respond to the query. The possible values are dependent on the value of the Destination Address. They are:

<u>Destination Address Value</u>	<u>SMK Id Value</u>
Individual Type 5 Device Address	Type 5 Device SMK
Individual Type 5 Device Address	Type 5 Device SMK and all Type "X" SMKs accessible through the addressed Type 5 device
Individual Type 5 Device Address	Type 5 Device SMK and all Type "X" SMKs on a specific Type "X" link attached to the addressed Type 5 device
Individual Type 5 Device Address	Specific Type "X" SMK accessible through the addressed Type 5 device
SMK Multicast Address	All Type 5 SMKs
SMK Multicast Address	All Type 5 SMKs and all Type "X" SMKs
SM Multicast	All HSE SMKs and all Type 1 SMKs on Type 1 link
Multicast Address used for the Annunciation Service	Configuration Applications (applications that maintain directory information)

Physical Device

This parameter is the Type 5 address to which the service request is to be sent. This address may be an Individual Type 5 Device Address, the SMK Multicast Address, or the Multicast Address used for the Annunciation Service.

Physical Device Tag

This selection type parameter, when present, contains the Physical Device Tag, indicating that the query is to locate a physical device.

S-VFD Tag

This selection type parameter, when present, contains the S-VFD Tag, indicating that the query is to locate a S-VFD.

S-VFD Reference

This selection type parameter, when present, contains a numeric reference to an S-VFD, indicating that the query is to locate a S-VFD.

Application Object Tag

This selection type parameter, when present, contains the tag of an application object, indicating that the query is to locate a tagged object.

Element ID

This conditional, user optional parameter is used only the Application Object Tag. It contains a reference to an element of the tagged application object.

11.2.2.3.1.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

11.2.2.3.2 Find Tag Reply Service

This unconfirmed service is used to send a reply to the initiator of a Find Tag Query. It contains information that can be used to establish an AR and an S-VFD Context to access the queried object.

This service operates over a pre-established client/server AR to the receiver whose address is specified by the Source Address parameter in the associated Find Tag Query.

11.2.2.3.2.1 Service Parameters

The service parameters for this service are shown in Table 303.

Table 303 – Find Tag Reply Service Parameters

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Invoke ID	M	M(=)
Queried Object Address		M
Queried Object Type "X" Device Address	C	C(=)
Queried Object Device ID	M	M(=)
Queried Object S-VFD Reference	C	C(=)
Queried Object S-VFD Version Number	C	C(=)
Queried Object Numeric Id	C	C(=)
List of S-VFD Context Conveyed References	C	C(=)

Argument

The argument contains the parameters of the service request.

Queried Object Address

This parameter is the Type 5 address to be used to access the queried object.

Queried Object Type "X" Device Address

This conditional parameter is present if the queried object is located in a Type "X" device on a Type "X" link that is connected to the responding Type 5 device. It contains the link and device address used to access the Type "X" device.

Queried Object Device ID

This parameter contains the value of the SMK attribute of the same name for the device that contains the queried object.

Queried Object S-VFD Reference

This conditional type parameter, when present, contains a numeric reference of the queried object's S-VFD. It is present for all queries except for Physical Device Tag queries.

Queried Object S-VFD Version Number

This conditional type parameter, when present, contains the version number of the queried object's S-VFD. This is the "Version of List" component of the S-VFD attribute "List Of Managed S-VFD Attributes". It is present for all queries except for Physical Device Tag queries.

Queried Object Numeric Id

This conditional type parameter, when present, contains the numeric id attribute of the queried object. It is present if the queried object is contained in a S-VFD.

List of S-VFD Context Conveyed References

This conditional type parameter, when present, contains a list of Conveyed References that can be used with the S-VFD Initiate Service to open a context to the queried object. It is present for all queries except for Physical Device Tag queries.

11.2.2.3.2.2 Service Procedure

The Unconfirmed Service Procedure specified in clause 4 applies to this service.

11.2.2.3.3 Identify Service

This confirmed service is used to obtain identification information from a remote SMK.

This service operates over dynamically established or pre-established client/server Type 5 ARs or over a publisher/subscriber AR. The remote address may be an individual Type 5 device address for client/server ARs, or the SMK multicast address or the multicast address used for the Annunciation Service for publisher / subscriber ARs. Each responding SMK returns its response using a pre-established client/server AR to the receiver whose address is specified by the Destination Address parameter.

11.2.2.3.3.1 Service Parameters

The service parameters for this service are shown in Table 304.

Table 304 – Identify Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Source Address		C		
Type "X" Address	C	C(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Type "X" Address			C	C(=)
Device ID			M	M(=)
Physical Device Tag			M	M(=)
Device Type			M	M(=)
SMK Multicast Address			C	C(=)
Annunciation Repeat Time			C	C(=)
Device Redundancy Capability			C	C(=)
Device Redundancy State			C	C(=)
Device Index			C	C(=)
Max Device Index			C	C(=)
SMK Version Number			C	C(=)
Device Version Number			C	C(=)
Version Number List			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Type "X" Address			C	C(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Type "X" Address

This conditional parameter contains the address of a Type "X" link connected to a Type 5 device, or the address of a device on a Type "X" link that is connected to a Type 5 device. It is present if the service is being requested of a Type "X" device SMK.

Result(+)

This selection type parameter indicates that the service request succeeded.

Type "X" Device Address

This conditional parameter contains the Type X address from which the service was requested. It may be the address of the Type "X" link connected to a Type 5 device, or the address of a device on a Type "X" link that is connected to a Type 5 device from which the service was requested. It is present if the service was requested from a Type "X" device.

Device ID

This parameter contains the value of the SMK attribute of the same name.

Physical Device Tag

This parameter contains the value of the SMK attribute of the same name.

Device Type

This parameter contains the value of the SMK attribute of the same name.

SMK Multicast Address

This parameter contains the value of the SMK attribute of the same name.

Annunciation Repeat Time

This parameter contains the value of the SMK attribute of the same name.

Device Redundancy Capability

This parameter contains the value of the SMK attribute of the same name.

Device Redundancy State

This parameter contains the value of the SMK attribute of the same name.

Device Index

This parameter contains the value of the SMK attribute of the same name.

Max Device Index

This parameter contains the value of the SMK attribute of the same name.

SMK Version Number

This parameter contains the value of the SMK attribute of the same name.

Device Version Number

This parameter contains the value of the SMK attribute of the same name.

Version Number List

This parameter is used by device that connects links of two or more Communication Model Types together.

If the Type "X" address request parameter is not present, then this parameter contains the link id and version number of each attached link. This Type "X" link version number is incremented each time any device address is added or deleted from the Type "X" network.

If the Type "X" address request parameter identifies a specific link connected to the receiving Type 5 device, then this parameter contains the Type "X" address and version number associated with each device connected to that link. This Type "X" device address version number is

incremented each time that specific device address is added or deleted from the Type "X" network.

Result(-)

This selection type parameter indicates that the service request failed.

Type "X" Address

This conditional parameter contains the Type X address from which the service was requested. It may be the address of the Type "X" link connected to a Type 5 device, or the address of a device on a Type "X" link that is connected to a Type 5 device from which the service was requested. It is present if the service was requested from a Type "X" device.

11.2.2.3.3.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

11.2.2.3.4 Annunciate Service

This unconfirmed service is used to multicast a notification onto the network that contains basic SMK information.

This service operates over a publisher/subscriber AR. The remote address for the AR is a multicast address defined for use with this service. The value for this address is specific to the underlying layer selected to support the socket services.

11.2.2.3.4.1 Service Parameters

The service parameters for this service are shown in Table 305.

Table 305 – Annunciate Service Parameters

Parameter name	Req	Ind
Argument	M	M(=)
AREP	M	M(=)
Device ID	M	M(=)
Physical Device Tag	M	M(=)
Device Type	M	M(=)
SMK Multicast Address	M	M(=)
Annunciation Repeat Time	M	M(=)
Device Redundancy Capability	M	M(=)
Device Redundancy State	M	M(=)
Device Index	M	M(=)
Max Device Index	M	M(=)
SMK Version Number	M	M(=)
Device Version Number	M	M(=)
Version Number List	M	M(=)

Argument

The argument contains the parameters of the service request.

Device ID

This parameter contains the value of the SMK attribute of the same name.

Physical Device Tag

This parameter contains the value of the SMK attribute of the same name.

Device Type

This parameter contains the value of the SMK attribute of the same name.

SMK Multicast Address

This parameter contains the value of the SMK attribute of the same name.

Annunciation Repeat Time

This parameter contains the value of the SMK attribute of the same name.

Device Redundancy Capability

This parameter contains the value of the SMK attribute of the same name.

Device Redundancy State

This parameter contains the value of the SMK attribute of the same name.

Device Index

This parameter contains the value of the SMK attribute of the same name.

Max Device Index

This parameter contains the value of the SMK attribute of the same name.

SMK Version Number

This parameter contains the value of the SMK attribute of the same name.

Device Version Number

This parameter contains the value of the SMK attribute of the same name.

Version Number List

This parameter is present if the service was requested by a device that connects links of two or more Communication Model Types together and if the Type "X" address request parameter is not present. This parameter contains the version number of each attached link.

11.2.2.3.4.2 Service Procedure

The unconfirmed Service Procedure specified in clause 4 applies to this service.

11.2.2.3.5 Device Assignment Service

This service is used to update the attribute values of the SMK addressed by the Destination Address parameter. The SMK receiving the new values updates its attributes, if possible, and returns a response that includes the newly updated attribute values. This service also may be used to assign a new address to a Type "X" device that is connected to a Type 5 device through a Type "X" network.

This service operates over dynamically established or pre-established client/server Type 5 ARs.

11.2.2.3.5.1 Service Parameters

The service parameters for this service are shown in Table 306.

Table 306 – Device Assignment Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Source Address		C(=)		
Type "X" Address	C	C(=)		
Device ID	M	M(=)		
Physical Device Tag	M	M(=)		
Device Type	M	M(=)		
SMK Multicast Address	C	C(=)		
Device Redundancy Capability	C	C(=)		
Device Redundancy State	C	C(=)		
Device Index	C	C(=)		
Max Device Index	C	C(=)		
Annunciation Repeat Time	S	S(=)		
New Type "X" Address	S	S(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Type "X" Address			C	C(=)
Device ID			M	M(=)
Physical Device Tag			M	M(=)
Device Type			M	M(=)
SMK Multicast Address			C	C(=)
Annunciation Repeat Time			C	C(=)
Device Redundancy Capability			C	C(=)
Device Redundancy State			C	C(=)
Device Index			C	C(=)
Max Device Index			C	C(=)
SMK Version Number			C	C(=)
Device Version Number			C	C(=)
Version Number List			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Type "X" Address			C	C(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Destination Address

This parameter is the address from which the service was requested. It is present if the AREP operates as pre-established.

Type "X" Address

This conditional parameter contains the address of a Type "X" link connected to a Type 5 device, or the address of a device on a Type "X" link that is connected to a Type 5 device. It is present if the service is being requested of a Type "X" device SMK.

Device ID

This parameter contains the value of the SMK attribute of the same name. This value must match that of the SMK attribute and is not writable.

Physical Device Tag

This parameter contains the value of the SMK attribute of the same name. This value is writable. Writing a new value causes the SMK to place all ASEs and ARs in an initialised state and causes all established ARs and S-VFD contexts to be abruptly terminated.

Device Type

This parameter contains the value of the SMK attribute of the same name.

SMK Multicast Address

This parameter contains the value of the SMK attribute of the same name. This value is writable.

Device Redundancy Capability

This parameter contains the value of the SMK attribute of the same name. It is present if the request is addressed to a Type 5 SMK.

Device Redundancy State

This parameter contains the value of the SMK attribute of the same name. It is present if the request is addressed to a Type 5 SMK.

Device Index

This parameter contains the value of the SMK attribute of the same name. It is present if the request is addressed to a Type 5 SMK.

Max Device Index

This parameter contains the value of the SMK attribute of the same name. It is present if the request is addressed to a Type 5 SMK.

Annunciation Repeat Time

This selection type parameter contains the value of the SMK attribute of the same name. This value is writable. It is selected if the request is addressed to a Type 5 SMK. During device assignment, the Type 5 FAL AE may negotiate this value to a larger value no larger than 5000 if the assigned value is less than 5000 and the device is not capable of supporting the assigned value.

New Type "X" Address

This selection type parameter contains a NULL value or an address that is to be assigned to a Type "X" device connected to the Type 5 device through a Type "X" network. It is selected if the Type "X" Address parameter is present.

Result(+)

This selection type parameter indicates that the service request succeeded.

Type "X" Address

This conditional parameter contains the Type X address from which the service was requested. It may be the address of the Type "X" link connected to a Type 5 device, or the address of a device on a Type "X" link that is connected to a Type 5 device from which the service was requested. It is present if the service was requested from a Type "X" device.

Device ID

This parameter contains the value of the SMK attribute of the same name.

Physical Device Tag

This parameter contains the value of the SMK attribute of the same name.

Device Type

This parameter contains the value of the SMK attribute of the same name.

SMK Multicast Address

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Annunciation Repeat Time

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Device Redundancy Capability

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Device Redundancy State

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Device Index

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Max Device Index

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK. The responder may negotiate this value to a smaller value, but no smaller than 500. That is, a value smaller than 500 can be returned only if that value or a smaller value is received in request.

SMK Version Number

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Device Version Number

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Version Number List

This conditional parameter is present if the service was requested from a device that connects links of two or more Communication Model Types together and if the Type "X" address request parameter is not present. This parameter contains the version number of each attached link.

Result(-)

This selection type parameter indicates that the service request failed.

Type "X" Address

This conditional parameter contains the Type X address from which the service was requested. It may be the address of the Type "X" link connected to a Type 5 device, or the address of a device on a Type "X" link that is connected to a Type 5 device from which the service was requested. It is present if the service was requested from a Type "X" device.

11.2.2.3.5.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service. The service fails if the Device ID and Device Type parameters do not match those of the addressed SMK.

11.2.2.3.6 Clear Address Service

This service is used to clear the address of a Type 5 device, a Type "X" link interface connected to a Type 5 device, or a device on a Type "X" link that is connected to a Type 5 device. After clearing the address of a Type "X" link interface connected to a Type 5 device or a device on a Type "X" link that is connected to a Type 5 device, the responding Type 5 device SMK returns the response. After accepting the request to clear its own address, the Type 5 device SMK returns the response, and then clears its address.

This service operates over dynamically established or pre-established client/server Type 5 ARs.

11.2.2.3.6.1 Service Parameters

The service parameters for this service are shown in Table 307.

Table 307 – Clear Address Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
Destination Address	M	M(=)		
Type "X" Address	C	C(=)		
Physical Device Tag	M	M(=)		
Result (+)			S	S(=)
AREP			M	M(=)
Invoke ID			M	M(=)
Destination Address			M	M(=)
Type "X" Address			C	C(=)
Device ID			M	M(=)
Physical Device Tag			M	M(=)
Device Type			M	M(=)
SMK Multicast Address			C	C(=)
Annunciation Repeat Time			C	C(=)
Device Redundancy Capability			C	C(=)
Device Redundancy State			C	C(=)
Device Index			C	C(=)
Max Device Index			C	C(=)
SMK Version Number			C	C(=)
Device Version Number			C	C(=)
Version Number List			C	C(=)
Result (-)			S	S(=)
AREP			M	M(=)
Destination Address			M	M(=)
Invoke ID			M	M(=)
Error Info			M	M(=)

Argument

The argument contains the parameters of the service request.

Destination Address

This parameter is the address to which the service request is to be sent. The request may be addressed to a Type 5 device, to a Type "X" link connected to a Type 5 device, or to a device on a Type "X" link that is connected to a Type 5 device.

Type "X" Address

This conditional parameter contains the address of a Type "X" link connected to a Type 5 device, or the address of a device on a Type "X" link that is connected to a Type 5 device. It is present if the service is being requested of a Type "X" device SMK.

Physical Device Tag

This parameter contains the value of the SMK attribute of the same name.

Result(+)

This selection type parameter indicates that the service request succeeded.

Destination Address

This parameter is the address to which the service response is to be returned.

Type "X" Address

This conditional parameter contains the Type X address from which the service was requested. It may be the address of the Type "X" link connected to a Type 5 device, or the address of a device on a Type "X" link that is connected to a Type 5 device from which the service was requested. It is present if the service was requested from a Type "X" device.

Device ID

This parameter contains the value of the SMK attribute of the same name.

Physical Device Tag

This parameter contains the value of the SMK attribute of the same name.

Device Type

This parameter contains the value of the SMK attribute of the same name.

SMK Multicast Address

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Annunciation Repeat Time

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Device Redundancy Capability

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Device Redundancy State

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Device Index

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Max Device Index

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

SMK Version Number

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Device Version Number

This conditional parameter contains the value of the SMK attribute of the same name. It is present if the service was requested from a Type 5 device SMK.

Version Number List

This conditional parameter is present if the service was requested from a device that connects links of two or more Communication Model Types together and if the Type "X" address request parameter is not present. This parameter contains the version number of each attached link.

Result(-)

This selection type parameter indicates that the service request failed.

Destination Address

This parameter is the address to which the error service response is to be returned.

11.2.2.3.6.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service.

In addition, the Device ID, Physical Device Tag, and Device Type parameter values must match the corresponding attribute values in the SMK of the device that is to have its address cleared request. If they do not match, a negative response is returned.

11.2.3 LAN Redundancy ASE

11.2.3.1 Overview

Type 5 devices can have two network interfaces that are used for LAN redundancy. These interfaces are referred to as Network Interface A and Network Interface B. They are managed by a user of the LAN Redundancy ASE referred to as the LAN Redundancy Entity.

The LAN Redundancy Entity exchanges redundancy status information with other LAN Redundancy Entities using the Diagnostic Message Service. The LAN Redundancy Entity of a Type 5 device selects which network interface to use to send other Type 5 APDUs based on this status information.

Each LAN Redundancy Entity submits a Diagnostic Message Service request primitive at periodic intervals and the LAN Redundancy ASE constructs two Diagnostic Message Service APDUs and sends them on two predefined Publisher AR Type 5 AR endpoints concurrently. Each LAN Redundancy Entity receives pairs of service indications from the other Type 5 devices on the network that participate in LAN Redundancy. This information is used to determine which network interface (A or B) is to be selected to send other Type 5 APDUs.

11.2.3.2 LAN Redundancy Class Specification

11.2.3.2.1 LAN Redundancy Formal Model

FAL ASE: LAN Redundancy

CLASS: LAN Redundancy

CLASS ID: Not used

PARENT CLASS: TOP

SYSTEM MANAGEMENT ATTRIBUTES:

1	(m)	Attribute:	Version
2	(m)	Attribute:	Number of Network Interfaces
3	(m)	Attribute:	Max Sequence Number Difference
4	(m)	Attribute:	LAN Redundancy Flags
5	(m)	Attribute:	Diagnostic Message Interval
6	(m)	Attribute:	Ageing Time
7	(m)	Attribute:	Diagnostic Message Send Addresses
8	(m)	Attribute:	Diagnostic Message Receive Addresses
9	(m)	Attribute:	Number of Diagnostic Messages Received
10	(m)	Attribute:	Number of Faults Detected
11	(m)	Attribute:	Number of Diagnostic Message Service Indications Missed
12	(m)	Attribute:	List of Network Interface Status
13	(m)	Constraint:	Crossed Cable Detection Enabled = TRUE
13.1	(m)	Attribute:	Number of Diagnostic Message Service Indications Missed

SERVICES:

1	(o)	OpsService:	Diagnostic Message

11.2.3.2.1.1 System Management Attributes

The following attributes are accessible through System Management. They are not otherwise accessible.

Version

This attribute specifies the version of this object. Each time the value of any of its other attributes changes, the version number is incremented by 1. Version number 0 indicates that this object has not been configured.

Number of Network Interfaces

Specifies the number of network interfaces on this device. A device may have one or two network interfaces. They are labelled Network Interface A and Network Interface B. (Network Interface B is only used if there are two network interfaces.)

Max Sequence Number Difference

This attribute defines the maximum acceptable difference between the sequence number parameters in a pair of Diagnostic Message Service indications received from a single sending device. When the difference exceeds the value of this attribute, a fault in the path from the network interface sending the lower sequence number is detected.

LAN Redundancy Flags

This attribute controls how the Diagnostic Message Service is used by the LAN Redundancy Entity. Its value may indicate one or more of the following:

- Single Multicast Message Transmission Interface Enabled. The transmission policy for all Type 5 services with multicast destination addresses except for the Diagnostic Message and SMK Annunciate services.
 - False Transmit on both network interfaces
 - True Transmit on one network interface
- Crossed Cable Handling Enabled.
 - False Do not detect crossed cables
 - True Detect crossed cables
- Single Multicast Message Reception Interface Enabled. The reception policy for all Type 5 services with multicast destination addresses except for Diagnostic Message and SMK Annunciate services.
 - False Listen for multicast addresses on both network interfaces
 - True Listen for multicast addresses on one network interface if zero or one fault detected
- Diagnosis using own messages enabled.
 - False Do not use own diagnostic messages for diagnosis
 - True Use own diagnostic messages for diagnosis

Diagnostic Message Interval

This attribute specifies the time interval in milliseconds between successive invocations of the Diagnostic Message Service by the LAN Redundancy Entity.

Ageing Time

This attribute specifies the time interval used by the LAN Redundancy Entity to remove silent devices from its list of devices that are participating in LAN redundancy. The value of this attribute must be larger than the value of the *Max Sequence Number Difference* attribute multiplied by the largest value of the parameter *Diagnostic Message Interval* found in Diagnostic Message Service indications.

Diagnostic Message Send Addresses

This attribute defines the destination addresses for sending Diagnostic Message APDUs. One address is defined for each network interface from which Diagnostic Message APDUs are sent. The addresses may be a broadcast, multicast or unicast addresses.

Diagnostic Message Receive Addresses

This attribute defines the addresses for receiving Diagnostic Message APDUs. One address is defined for each network interface on which Diagnostic Message APDUs are received. The addresses may be a broadcast, multicast or unicast addresses.

Number of Diagnostic Messages Received

This attribute counts the number of Diagnostics Message Service indications received (from all devices). A separate count is maintained for each network interface.

Number of Faults Detected

This attribute is a count of the total number of faults detected from missed Diagnostics Message Service indications. A separate count is maintained for each network interface.

Number of Diagnostic Message Service Indications Missed

This attribute is a count of the number of missed Diagnostics Message Service indications (from all devices) as computed from sequence number gaps. A separate count is maintained for each network interface.

List of Network Interface Status

This attribute is a list. Each entry in this list represents the status for the path to a remote device from which Diagnostic Messages Service indications are being or have been received. A separate list is maintained for each network interface on which Diagnostic Messages Service indications are received. The status for each remote device indicates whether or not Diagnostic Messages Service indications are being successfully received or whether the path is in a fault state (a fault has been detected and has not cleared).

List of Crossed Cable Status

This conditional attribute is a list. This attribute is present only if the value of the *Crossed Cable Detection Enabled* component of the LAN Redundancy Flags attribute is TRUE. Each entry in this list represents the Crossed Cable Status for the path to a remote device from which Diagnostic Messages Service indications are being or have been received. The Crossed Cable Status indicates whether the network interface used by the Diagnostic Message Service for requests, as indicated by the *Network Interface used for Transmission of this Diagnostic Message* parameter in the Diagnostic Message Service, is the same network interface used by the reporting device to receive the messages.

11.2.3.3 LAN Redundancy ASE Service Specification

This subclause contains the definition of the service that is unique to this ASE. The service defined for this ASE is:

Diagnostic Message Service

11.2.3.3.1 Diagnostic Message Service

This service operates only over a pair of publisher/subscriber ARs. At the publisher endpoint, one AR is bound to network interface A and the other AREP is bound to network interface B. Each service invocation causes Diagnostic Message Service APDU to be sent simultaneously from each.

11.2.3.3.1.1 Service Primitives

The service parameters for this service are shown in Table 308.

Table 308 – Diagnostic Message Service

Parameter name	Req	Ind
Device Index	M	M(=)
Max Device Index	M	M(=)
Number of Interfaces	M	M(=)
Diagnostic Message Interval	M	M(=)
List of Network Interface Status	M	M(=)
List of Crossed Cable Status	M	M(=)

Device Index

This value is obtained from the SMK attribute of the same name.

Max Device Index

This value is obtained from the SMK attribute of the same name.

Number of Network Interfaces

This value is derived from the SMK List of Network Addresses attribute. There is one network address per network interface in this list.

Diagnostic Message Interval

This value is obtained from the LAN Redundancy attribute of the same name.

List of Network Interface Status

This value is obtained from the LAN Redundancy attribute of the same name.

List of Crossed Cable Status

This value is obtained from the LAN Redundancy attribute of the same name.

11.3 Type 5 ARs

11.3.1 Publisher/Subscriber Type 5 AR Endpoint Formal Model

11.3.1.1 Description

This class is defined to support the on-demand queued distribution of unconfirmed services to one or more application processes. It is the Type 5 counterpart to the Type 1 QUU and BNU AREPs. The behaviour of this type of AR can be described as follows.

An AR ASE user wishing to convey an unconfirmed APDU submits an AR ASE Service Data Unit to the sending endpoint of the AR. The AREP sending the request APDU submits it to the underlying layer for transfer. The underlying layer sends it at its next opportunity. The AREP receiving the request APDU from its underlying layer delivers it to the AR ASE user in the order that it was received.

The sending endpoint has a role of PUBLISHER and the receiving endpoints have the role of SUBSCRIBER. PUBLISHER endpoints correspond to the Type 1 SUBSCRIBER and REPORT SOURCE endpoints. SUBSCRIBER endpoints correspond to the Type 1 SUBSCRIBER and REPORT SINK endpoints.

The following summarises the characteristics of this AREP class.

Roles: PUBLISHER (REPORT SOURCE¹)
SUBSCRIBER (REPORT SINK²)

Cardinality: 1-to-n

Timeliness: YES³

Notes: REPORT SOURCE is the same as PUBLISHER

REPORT SINK is the same as SUBSCRIBER

This AR can be configured to transparently convey a timeliness parameter for the FAL User.

11.3.1.2 Formal Class Definition

FAL ASE:	AR ASE
CLASS:	Publisher/Subscriber Type 5 AR Endpoint
CLASS ID:	Not Used
PARENT CLASS:	AR Endpoint

NETWORK MANAGEMENT ATTRIBUTES:

1. (m) Attribute: Role (PUBLISHER, SUBSCRIBER)
2. (m) Attribute: AREP State (OPEN, CLOSED)
3. (m) Attribute: Socket Mapping Reference

SERVICES:

1. (o) OpsService: Unconfirmed Send

11.3.1.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

PUBLISHEREndpoints of this type send messages to multiple subscriber endpoints using unconfirmed services operating over connectionless socket services. They send messages to a single subscriber endpoint using unconfirmed services operating over connection-oriented socket services. If connection-oriented services are used, then the publisher is the listener for the connection.

SUBSCRIBEREndpoints of this type receive reports from source endpoints using unconfirmed services operating over either connectionless or connection-oriented socket services. If connection-oriented services are used, then the subscriber is the initiator of the connection.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

Socket Mapping Reference

This attribute is a reference to the socket mapping to the underlying layer. These mappings are specified in IEC 61158-6.

11.3.1.2.2 Services

Unconfirmed Send

This optional service is used to send an unconfirmed service on an AR.

11.3.2 Client / Server AR Endpoint Formal Model

11.3.2.1 Description

This class is defined to support the on-demand exchange of confirmed and unconfirmed services between two application processes. It is the Type 5 counterpart to the Type 1 QUB Co AREP. The behaviour of this type of AR can be described as follows.

An AR ASE user wishing to convey a request or response APDU submits it as an AR ASE Service Data Unit to its AREP. The AREP sending the request APDU queues it to its underlying layer for transfer at the next available opportunity. Either endpoint may send unconfirmed request or response APDUs. Only the Client AREP may send the Establish Request APDU.

The AREP receiving the APDU from its underlying layer, queues it for delivery to its AR ASE user in the order that it was received.

The following summarises the characteristics of this AREP class.

Roles:	Client
	Server
Cardinality:	1-to-1
Timeliness:	No

11.3.2.2 Formal Class Definition

FAL ASE:	AR ASE
CLASS:	Client / Server AREP
CLASS ID:	Not Used
PARENT CLASS:	AR Endpoint

NETWORK MANAGEMENT ATTRIBUTES:

1. (m) Attribute: Role (CLIENT, SERVER)
2. (m) Attribute: AREP State
3. (m) Attribute: Socket Mapping Reference
4. (c) Constraint: Role = CLIENT
- 4.1 (m) Attribute: Server Physical Device Tag

SERVICES:

1. (o) OpsService: Confirmed Send
2. (o) OpsService: Unconfirmed Send
3. (o) OpsService: Establish

11.3.2.2.1 Network Management Attributes

Role

This attribute specifies the role of the AREP. The valid values are:

CLIENTEndpoints of this type issue confirmed service Request-APDUs to initiate the establishment of the AR. One and only one of the AREPs involved in an AR must be the initiator.

SERVEREndpoints of this type respond to requests received from the CLIENT to establish the AR.

AREP State

This attribute specifies the state of the AREP. The values for this attribute are specified in IEC 61158-6.

Server Physical Device Tag

This conditional attribute specifies the Physical Device Tag SMK attribute of the server device. This attribute is used to locate the server on the network. How the server is located is beyond the scope of this standard. This attribute is present only for CLIENT AREPs.

11.3.2.2.2 Services

Unconfirmed Send

This optional service is used to send an unconfirmed service on an AR.

Confirmed Send

This optional service is used to send a confirmed service on an AR.

Establish

This service is used to establish an AR.

11.4 Summary of FAL Type 1 and Type 5 Classes

Table 309 contains a summary of the defined FAL Classes. The Class ID values are defined in the corresponding clause of Type 1.

Table 309 – FAL Class Summary

FAL ASE	Class	Class ID
S-VFD	S-VFD	Not used
	S-VFD Context	Not used
Data Type	Fixed Length & String Data Type	See Type 1
	Structure Data Type	See Type 1
	Array Data Type	See Type 1
Type 1 AR	Client / Server	Not used
	Publisher / Subscriber	Not used
Type 1 Variable	Fixed Length & String Variable	See Type 1
	Array Variable	See Type 1
	Data Structure Variable	See Type 1
	Variable List	See Type 1
Type 1 Event	Event	See Type 1
	Notifier	See Type 1
Type 1 Load Region	Load Region	See Type 1
Type 1 Function Invocation	Function Invocation	See Type 1
SMK	SMK	Not used
LAN Redundancy	LAN Redundancy	Not used

11.5 Permitted FAL Type 1 and Type 5 Services by AREP Role

Table 310 below defines the valid combinations of services and AREP roles (which service APDUs and AREP with the specified role can send or receive). The Unc and Cnf columns indicate whether the service listed in the left-hand column is unconfirmed (Unc) or confirmed (Cnf).

Table 310– Services by AREP Role

	Unc	Cnf	Client	Server	Publ	Subsc
FAL Services			req rcv	req rcv	req rcv	req rcv
Type 1 Mgt ASE						
Create		X	X X	X X		
Delete		X	X X	X X		
Get Attributes		X	X X	X X		
Get Attribute List		X	X X	X X		
Set Attributes		X	X X	X X		
Begin Set Attributes		X	X X	X X		
End Set Attributes		X	X X	X X		
S-VFD ASE						
Identify		X	X X	X X		
Get Status		X	X X	X X		
Status Notification	X		X X	X X	X	X
Initiate		X	X X	X X		
Terminate		X	X X	X X		
Type 1 AR ASE						
AR-Confirmed Send			X X	X X		
AR-Unconfirmed Send			X X	X X	X	X
AR-Establish			X	X		
AR-Abort			X X	X X	X	X X
Type 1 Variable ASE						
Read		X	X X	X X		
Write		X	X X	X X		
Information Report	X		X X	X X	X	X
Type 1 Event ASE						
Confirmed Ack Event		X	X X	X X		
Enable Event		X	X X	X X		
Event Notification	X		X X	X X	X	X
Type 1 Load Region ASE						
Initiate Load		X	X X	X X		
Push Segment		X	X X	X X		
Pull Segment		X	X X	X X		
Terminate Load		X	X X	X X		
Type 1 Function Invocation ASE						
Start		X	X X	X X		
Stop		X	X X	X X		
Resume		X	X X	X X		
Reset		X	X X	X X		
Kill			X X	X X		
LAN Redundancy ASE						
Diagnostic Message	X				X	X
SMK ASE						
Find Tag Query	X		X X	X X	X	X
Find Tag Reply	X		X X	X X	X	X
Identify		X	X X	X X		
Annunciate	X				X	X
Device Assignment		X	X X	X X	X	X
Clear Address		X	X X	X X	X	X

12 Communication Model Type 6 Specification

There are no specific Type 6 application layer ASEs or ARs specified. Any FAL Communication Model that meets the Type 6 DLL interface specified in IEC 61158-3 can be used.

13 Communication Model Type 7 Specification

13.1 Type 7 Concepts

13.1.1 AL Type 7 environment

This general model presents the relationship between AL type 7 environment and the application layer services which is the purpose of this standard.

AL Type 7 environment is both process control and discrete parts manufacturing. In those systems, manufacturing devices are controlled by a system that performs the automation functions.

13.1.2 Application services

Application services can be divided into two sets.

The standard specifies the MPS (Manufacturing Periodic/apperiodic Services) application service set. The other service set corresponds to a subset (subMMS) of that specified in the MMS standard.

MPS services are particularly well suited to the real time requirements of a distributed application in the fact that they support:

- The periodic broadcast update of a distributed database in the various control system devices.
- The elaboration of temporal qualifiers associated with the data base.
- The elaboration of consistency qualifiers associated with sets of data within this database.

SubMMS and MCS are essentially considered to fulfil the requirements for configuration and operating mode management within the distributed application.

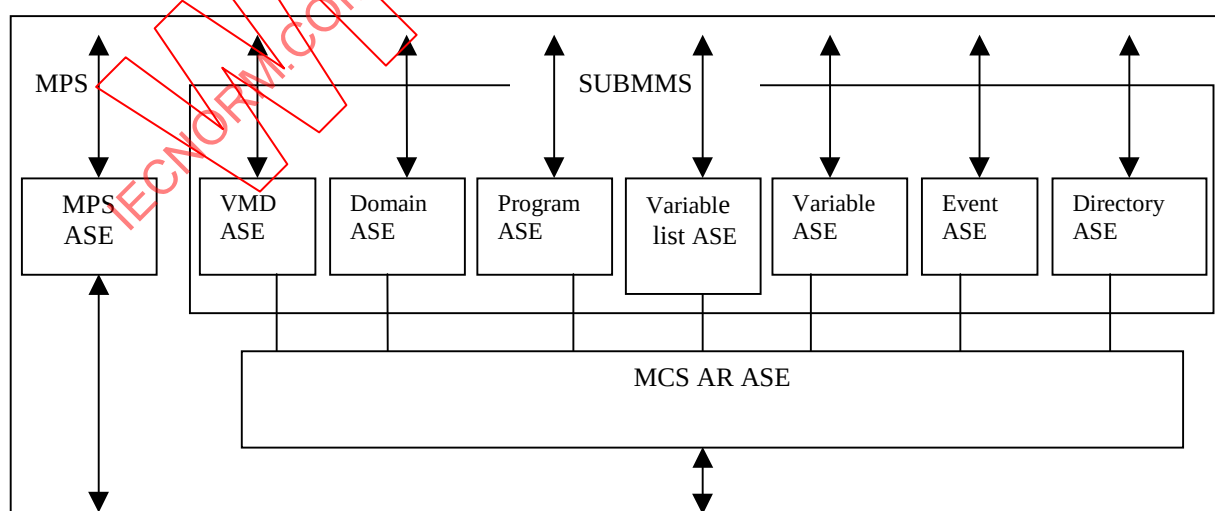


Figure 72 – Organisation of the type 7 ASEs and ARs

A distributed application is characterized by various information processing tasks on various devices of a communication system for the requirements of check a production system control application.

Cooperation between the tasks belonging to different devices is modelled in the form of interactions between application processes (APs).

A distributed application is composed of two or more APs, each of the devices of the communication system which can have zero, one or several of them.

As provided for by this standard, the communication between two APs passes via the network.

NOTE The breaking down of a distributed application in several AP's at the level of a device is decided by the user according to his own criteria (Methodology, Functional structuring of the application, Setting into service facilities, etc.)

13.1.3 Application process

An AP is modelled by one or several information processing tasks in a device for the requirements of a particular function of the distributed application.

The aspects of an AP, taken into account by AL Type 7, only concern its communication capacities which are modelled by Application Entities (AEs). Each AP can have at the most one AE of a given type.

On the other hand, the cooperation between APs in the framework of a distributed application are performed via the setting up of the relations between AP invocations (APIs) which represent its activities. An AP can, at a given instant, zero, one or several APIs. Besides, each API has one particular utilization of the AE (AEI) which satisfies the communication requirements.

13.1.3.1 AP model

The purpose of the AP model is to show all the aspects of an AP which are taken into account by the AL Type 7 communication.

NOTE The other aspects concerning the type of information, processing and execution of the latter are not modelled, since they are not involved at the communication level.

Class: APPLICATION PROCESS (AP)

Key Attribute: AP title

Attribute: AP type

Attribute: List of Inherit from Application Entity

Attribute: List of Reference API

13.1.3.1.1 Attributes:

AP title:

A title is attributed to an AP to identify it, among all the other APs, in a non-ambiguous manner in a multisegment AL Type 7 network. For the requirements of the universal identification of an AP in the AL Type 7 environment, this AP tile is involved as a constituent of the ISO name proper to AL Type 7 user site.

This identification can be structured from the name of the device to which it belongs.

NOTE To structure this identification, this AP must remain permanently in the equipment.

AP type:

The type of an AP helps to designate a set of APs to which it belongs.

NOTE This type is not covered by a title which can be handled in the AL Type 7 environment.

List of Inherit from Application Entity

The Application Process class inherits an Application Entity class list which gives it the capacity to communicate in the AL Type 7 environment.

Thus an AP has one or several application entities, which are necessarily of different types.

List of Reference API:

Designates the lists of APIs which model the AP activities. This list can vary dynamically but at a given instant zero, can designate one or several APIs.

13.1.3.2 API model

An API model describes the activities of an AP.

Class: APPLICATION PROCESS INVOCATION (API)

Key Attribute: API identification

Attribute: Reference Application Process

Attribute: List of Reference AEI

13.1.3.3 Attributes:

API identification:

Identifies an Application Process Invocation in a non-ambiguous manner in the given AP field. It must be underlined that the scope of this identification is global for the AP and independent of the rules for naming the AP title.

Reference Application process

The purpose of this attribute is to show that this API belongs to a given AP..

The lifetime of an API is necessarily less than or equal to that of the AP to which it belongs.

List of Reference AEI:

This list designates AE uses which are reserved for the API requirements.

An API can use zero, one or several AEIs for each AE of the AP to which it belongs.

The number of referenced AEIs can vary dynamically in time. Consequently, an API, at a given moment, can have one or several AEIs (Complete definition of an AEI in the following paragraph).

13.1.4 Application entity

An AE represents a set of communication capacities for the requirements of an AP. These communication capacities are defined by a set of Application Service Elements (ASEs). The particular uses of the communication capacities of an AE are represented by AE invocations (AEIs).

An AEI consequently models the communication functions and their state for the particular activities of an API. Communications between APIs take place via AEIs which, as required, may or may not be related by Application Associations (AAs) or Application Relationship (AR) in the AL TYPE 7 context.

An AEI in the AL TYPE 7 environment gives an API the exclusive possibility of communicating with or without association. An AEI can only participate in one and in only one association application.

The communications without association allow an AEI to exchange information with one or several other AEIs.

An association application allows an AEI to communicate with one or several other AEIs in the framework of a common Context Application (CA).

This context can be negotiated or predefined by configuration for each AEI, for a point to point association which makes use of a pair of AEIs. The context is necessarily prenegotiated for each AEI.

The state of an AEI is thus directly linked to the fact that the latter participates or not in an association and to the state of the association if it participates in a negotiated association.

Generally it can be said that the lifetime of an AEI is equivalent to that of the service exchanged for the communications without association and equivalent to that of the association for communications with associations. However, for a negotiated association, the lifetime of the pair of AEIs is lengthened by their opening and closing transients.

The lifetime of an AEI is controlled by the API which it represents in the AL TYPE 7 environment. An API, on the other hand, can have a much longer lifetime than that of its AEIs, equal to that of the AP to which it belongs.

Particular AR are predefined by configuration, these AR are only used by the MPS ASE to exchange plain data (variable). All the parameters relative to the variable are pre-established at the configuration, these parameters never change. Only a new configuration can cancel or modify these particular AR. The identification of this peculiar AR is made by a number unique on the network, this number refer to the whole set of parameters of the AR and the DLL resources implied for the communication. This number similar to a DLCEP Id is called Identifier.

13.1.5 AE model

Description of the communication capacities offered by an AE.

Class: APPLICATION ENTITY (AE)

Attribute: AE type

Key Attribute: Qualifier

Attribute: List of Inherit from ASE

Attribute: List of Reference AEI

13.1.5.1 Attributes:

Qualifier:

An AE qualifier identifies an AE non-ambiguously in the field of an application process. The combination of this AE qualifier with the AP title forms an AE title which identifies this AE non-ambiguously in the AL Type 7 environment.

This AE title allows recovery from the directory of the addressing information, the link addresses.

An AE title can have "Synonyms of the AE title" to be distinguished from one or several other AEs by different titles.

An AE can be globally identified with other AEs with the help of an "AE group designation".

AE type:

The AE type designates a set of AEs sharing the same communication characteristics. Various AEs of the same type inherit the same list of application service elements.

List of Inherit from ASE

The Application Entity (AE) class inherits an ASE class list which gives it the specificity of its communication functions. An AE can support one or several ASEs, the latter being necessarily of different types.

List of Reference AEI

Designates the list of AEIs which model the AE uses. This list, which can vary dynamically, can designate at a given moment, zero, one or several AEIs.

13.1.6 AEI model

An AEI model is used to describe a particular use of the AE.

Class: APPLICATION ENTITY INVOCATION (AEI)

Key attribute: AEI identification

Attribute: Reference AE

Attribute: Reference API

Attribute: Local AE address

Attribute: Remote AE address

Attribute: Communication mode

[ASSOCIATED; NON-ASSOCIATED]

Constraint: Communication mode = ASSOCIATED

Attribute: Association state

[OPENING;

ESTABLISHED;

CLOSED]

Attribute: Association type

[NEGOTIATED; PRENEGOTIATED]

Attribute: Association type

[NEGOTIATED; PRENEGOTIATED]

Attribute: Local AEI access point

Attribute: Remote AEI access point

Attribute: Inherit from: Application context

13.1.6.1 Attributes:

AEI identification:

Allows non-ambiguous identification of an application entity invocation in the sub-field of an AE reserved for an API of a given AP.

The Non-ambiguous designation of an AEI in the environment requires indicating: AP title / API identification / AE qualifier / AEI identification.

Reference AE:

This reference is used to designate the AE object to which this AEI belongs,

The lifetime of an AEI is necessarily less than or equal to that of the AE to which it belongs.

Reference API:

This reference helps to designate the API object which is represented in the AL Type 7 environment by this

AEI. An AEI can represent one and only one API in the AL Type 7 environment.

Local AE address:

Corresponds to the link address which allows locating the AE to which this AEI belongs.

This address is completed with the help of directory functions during the creation of this AEI and remains static for its lifetime.

Remote AE address:

Corresponds, either to a link address which allows locating the AE with which this AEI is corresponding, for point to point exchanges, or with a link address which helps to locate an AE group with which this AEI is corresponding for multipoint exchanges.

This address is filled at the AEI level during the opening of an association with one other AEI or when exchanging data in a non-associated mode.

Communication mode [ASSOCIATED; NON-ASSOCIATED]

(The communication mode in the ASSOCIATED state indicated that this AEI allows an API to communicate in a application context which was previously defined or negotiated. The type of exchanges between AEIs, possible in such a communication mode, is point to point or multipoint.

When the communication mode is NON-ASSOCIATED, this AEI allows an API to communicate outside all pre-established context. The nature of the possible exchanges in a non-associated communication mode is point to point or multipoint.

Association state

[OPENING;

ESTABLISHED;

CLOSING];

In the associated mode, the possibilities of communication of an AEI with another AEI depends on the state of the association. The various states correspond to the phases in the life of an association application:

OPENING:

Corresponds to an AEI which, at the request of the API which it represents, is negotiating an application context with another AEI, for the requirements of an association. In this state, no exchanges can take place outside those reserved for negotiation of the context.

ESTABLISHED:

Corresponds to an AEI which offers the API which it represents, the possibility of communicating with another API via one of its determined AEIs, within the framework of an application context. This application context has been obtained either by negotiation between the AEI pair or by local predefinition at the AEIs.

CLOSING:

Corresponds to an AEI which, at the request of the API which it represents, or with which it had been associated, is being freed from its application context.

Association type

[NEGOTIATED;
PRENEGOTIATED]

This type indicates if the association has been negotiated or not. This attribute, if it takes the PRENEGOTIATED value, imposes on the association to remain exclusively in the ESTABLISHED state.

Local AEI access point:

Each AEI, in the ASSOCIATED mode, can be directly localized by its AEI access point. This addressing information is filled with the help of the directory functions during the creation of an AEI instance and remains static for its full lifetime.

AEI remote access point:

Each AEI, which is in the ASSOCIATED mode, has this addressing information which localizes the AEI with which it is in correspondence for point to point exchanges. This information is filled in for an AEI at the opening of an association.

This information locates the AEIs with which it is in correspondence for multipoint exchanges.

Inherit from Context Application

This inheritance allows designating the object of the application context which conditions communications with this AEI.

The attributes of this application context are valued during installation of an AEI on the basis of the values proposed by the user in the negotiated associated mode and the non associated mode, whereas they are filled by configuration in the prenegotiated associated mode.

13.1.7 Application context

A pair of AEIs must have common knowledge of the rules which govern their communications. This set of rules is called Context Application. The application context which can be supported by an AEI is necessarily derived from the possibilities offered by the various ASEs possessed by the AE to which it belongs.

13.1.7.1 Application Service Elements

The possibilities offered by the various ASEs of an AE are defined by the specification (in ASN1) of one or more set of Application Protocol Data Units (APDU) which form abstract syntaxes. Besides, the use of an abstract syntax to dialogue between two AEs requires the implementation of a transfer syntax which defines the rules for encoding of the APDUs on the bus.

NOTE Among the abstract syntaxes associated with the ASE MMS, the general abstract syntax ISO-9506-MMS-1 can be noted as well as other abstract syntaxes such as those meant for digital controls ISO-9506-NCCS-1 and process control ISO-9506-PROCESS-1.

Class: ELEMENT SERVICE APPLICATION

Attribute: ASE type

Attribute: List of Abstract syntax

Attribute: List of Transfer syntax

13.1.7.1.1 Attributes:

ASE type:

Several ASE types can be met with in the AL TYPE 7 - environment. This standard is covered by the SubMMSE definition derived from the ASE MMS defined by the ISO.

The MCSE (Messaging Common Service Element) service corresponds to the control ASE which must necessarily be used to project another ASE application on the link layer messaging. This service element has unique encoding rules which are universally known in the AL TYPE 7 - messaging environment.

NOTE The MCS encoding rules are not given in ASN1. They are directly given in the representation used for the transfer.

List of Abstract Syntax:

An application service element can have one or several abstract syntaxes. Each of them correspond to a set of APDUs described in an ASN1 module universally identified in the AL TYPE 7 - environment.

NOTE For example, an abstract syntax defined in ASN1 has the following appearance:

```
<module name> <module universal identifier>
Definitions::= BEGIN{
IMPORTS <..>, <..>, ....FROM <external module>
EXPORTS <..>, <..>, ...
    PDU module ::= CHOICE {
        {<production name> [N°] <production>} +
    }
END
```

This standard keeps AL TYPE 7 - SUBNMMS-1, derived from the reference core, as one of its abstract syntaxes for the MMS service element in AL TYPE 7 - .

List of Transfer syntax:

A transfer syntax defines the abstract syntax encoding rules used. An ASE could support several transfer syntaxes within the framework of an AE, which could be used according to the requirements depending on the possibilities of their correspondents.

One of the transfer syntaxes taken into account by this AL Type 7 environment standard for the implementation of the AL Type 7 abstract syntax, AL Type 7 SUBMMMS-1, is proposed by the ISO basic encoding rules (BER).

Like the abstract syntaxes, the transfer syntaxes are universally identified in the AL TYPE 7 - environment.

13.1.7.2 Type of application context

The application context belonging to an AEI conditions the possible communications with it. In the information contained in the application context, some is specific to the AE architecture, i.e. with the type of ASE resting on MCS as well as with the abstract transfer syntax selected. Other information is specific to the choice performed on the API level and corresponding to the restrictions to the negotiated abstract syntax and to the quantity of communication resources used.

Class: APPLICATION CONTEXT

Attribute: Context name

Key Attribute: Context identification

Attribute: Reference ASE

Attribute: Abstract syntax

Attribute: Transfer syntax

Constraint: Com. mode = ASSOCIATED

Attribute: Application reductions

Attribute: Service quality

13.1.7.2.1 Attributes:**Identification of the context:**

Allows identifying an application context in the field of an AE. It thus has an exclusive local role.

Context name:

Used to identify, in the AL TYPE 7 - environment, the bases of a context consisting of the selected ASE, an abstract syntax and a transfer syntax. This name is used during the association negotiation or during a transfer of data in the non-associated mode.

Reference ASE:

One of the AE application service elements which is selected by this application context.

Abstract syntax:

An abstract syntax among those of the ASE within the framework of the AE, which is selected by this context.

Transfer syntax:

The transfer syntax selected for encoding the ASE PDUs, implemented within the framework of this AE, which is selected by this context.

Application reductions:

The abstract syntax selected at the level of an AEI application context for a given service element can be used in a reduced manner. This possibility is used only for transmission in the associated mode.

Service quality:

The selected service quality.

13.1.7.3 Directory

In order to communicate mutually, the AEIs make use of directory functions which given them the addressing information which they require. These directory functions are necessary exclusively during the opening of association or during a transmission in the mode without association.

13.1.7.3.1 Addressing model

The link layer offers data transfer services in the non-connected mode and an addressing space allowing location of the application entities. This addressing space allows locating the entities individually and/or by groups, and on the other hand to locate them with the help of physical or logic addresses. These two types of addresses respectively allow taking into account or not of the network topology in the location of application entities.

The application layer has a double location requirement, that of the AEs during negotiation of the associations or transmission of data without association, and that of the AEIs during transmission of data within the framework of an association.

Consequently, the AE addresses and the AEI access points which must be located are addressed with the link addresses from the single link addressing space.

In the case of point to point exchanges, the AE addresses and the AEI access points are designated with the help of individual link addresses. In the case of multipoint exchanges, the AE address and the AEI access point locating the transmitter are designated by individual link addresses whereas the AE address and the AEI access point locating the receivers are designated by group link addresses.

The rules allocating the liaison addresses used for the AE addresses and the AEI access points are not defined in this standard.

The abbreviations used to speak about these two uses of link addresses are: ADAE for those which are associated with AE addresses and ADAEI for those which are associated with AEI access points.

NOTE The ADAEIs are allocated by the directory functions of a device to the AEIs, in the sub-set which was attributed to an AE.

The address values contained in this sub-set can be, for example selected by dividing up the addressing space according to the network architecture or well chosen according to other criteria better satisfying the communication requirements of the user application.

Addressing rules:

Rule_1: Each AE is identified by one and only one "AE title" which is in a bi-univocal relation with an ADAE, allowing it to be located in the AL TYPE 7 - environment.

Rule_2: Each AE can be designated by one or several synonyms, each being in relation with one and only one address.

Rule_3: Each AE can be designated in the framework of one or several "AE group designations", each being in relation with one and only one address.

NOTE It is important to note that biunivocity is not a property of rules 2 and 3.

Rule_3: Each AEI which is identified by an AEI identification in the field of an AE, an API, of a given AP, is in biunivocal relation with an ADAEI, allowing it to be located in the AL TYPE 7 - environment.

13.1.7.3.2 Directory functions

The projection between the names allowing identification of the various objects (AEs and AEIs) in the application, and the link addresses (respectively ADAE and ADAEI) used to locate them in the AL TYPE 7 - environment, is managed by the directory service.

For any communication initiative at the level of an AEI, the latter calls the directory functions which give it the addresses necessary for the exchanges.

- Initiator Directory Function (IDF):

This function is implemented for an AEI initiating an exchange concerning an association opening or a transmission of data in the non-associated mode.

IDF is used to:

- obtain an ADAE pair allowing location of the local and remote AEs according to their AE titles,
- allocate an ADAEI for the local AEI access point and optionally of another for the remote AEI access point. This allocation only takes place when setting up an association.

<Input parameters> ::=

<Local AE title>
 <Remote AE title>
 | Synonym of the AE title
 | Designation of the AE group>
 <Transmission mode>

with <Transmission mode>::=<ASSOCIATED | NON-ASSOCIATED>;

<Output parameter> ::=

<Local ADAE>
 <Remote ADAE>
 [<Local ADAEI>] (*)
 [<Remove ADAE>] (**)

(*) This ADAEI is only returned in the case of an "ASSOCIATED" transmission mode.

(**) This ADAEI is returned or not according to the remote AE title.

Responder Directory Function (RDF):

This function is implemented at the level of a called AE, for the opening of an association.

RDF:

Used to allocate an ADAE at the level of the called entity during the response phase for an association opening request.

<Input parameters> ::=

<Local ADAE>
 <Remote ADAE>

<Output parameters> ::=

<Local ADAEI>

NOTE This function is not used on the level of a responder if the initiator of the communication had already performed the allocation by use of the function.

13.1.8 Coordination of the AEs

The initiative and the type of communications belong to the APIs. The four types of communication are:

- Information exchanges in the non-associated mode.

- Association opening requests.
- Information exchanges in the associated mode.
- Association closing requests

NOTE For each of these types of communication, the AL TYPE 7 - environment objects are implemented in a specific manner which is covered by the following paragraphs.

13.1.8.1 Information exchanges in the non-associated mode

When an API wishes to initialize a data transfer in the non-associated mode, it proceeds in the following manner:

Exchange scenario:

- It addresses, as applicable:
 - either the source AE identified by its <AE title>, indicating the addressee (either by its <AE title>, or by an <AE synonym title>, or by an <AE group designation> and the <Application context>,
 - or an AEI whose mode is NON-ASSOCIATED, if the latter has been created to allow a series of exchanges.
- The local procedure which takes place at the AE after this request by the API is as follows:

If the API addresses the AE, the following local procedure is executed:

 - Creation of an AEI object in the NON-ASSOCIATED mode.
 - Entering of the attributes of this object:
 - <Application context> takes the value indicated by the request,
 - <Remote AE address> takes the value returned by IDF for the non-associated communication mode,
 - <Local AE address>: takes the value returned by ID for the non-associated communication mode.
 - Then whatever the addressing which took place, a data transfer service in the MCSE non-associated mode takes place within the framework of this AEI. The user data encapsulated in this data transfer is that of the MCS user application service which was requested.
 - The AEI thus used could be maintained or revoked after this exchange. In case of revocation, this will be effective immediately if the service is not confirmed or after reception of the response and passage to API of the data contained in it, if the service is confirmed.
- The following procedure is executed at the addressee AE:
 - Depending on whether an AEI does not yet exist or exists already, the following operations are executed or not:
 - Creation of an AEI object in the NON-ASSOCIATED mode.
 - Filling of the attributes of this object:
 - <Application context>: takes the value indicated in the received PDU, if it is supported by the AE.
 - <Remote AE address> takes the value contained in the received PDU.
 - <Local AE address> takes the value contained in the PDU.
 - Identification of an API object designated implicitly or explicitly in the PDU.
 - Passage of the <Application service> contained in the PDU received at the API identified for the execution.
 - The AEI thus used can be revoked or preserved at the end of this exchange. In case of revocation, this will be effective immediately if the service is not confirmed or after transmission of the response if it is confirmed.

13.1.8.2 Association opening request

If the API wishes to initialize an association opening request, it proceeds in the following manner:

Opening scenario:

- The API locally addresses the calling AE identified by its <AE title>, indicating:
 - the called <AE title>,
 - the <opening service> (Initiate Rq in MMS) which contains the <Application reductions> of the association, which will be negotiated,
 - the <Application context> description other than the application reductions which will be negotiated at the MCS level.
- The local procedure which will be executed at the AE level after this API request, is the following:
 - Creation of an AEI object in the ASSOCIATED mode in the OPENING state.
 - Entering of the attributes of this object:
 - <Remote AE address; local AEI access point>: take the value returned by IDF.
 - Consideration of the application context requested by the user, and possible reduction of this, if its local AE cannot offer it.
 - A_Associate.Rq service call by MCSE in the framework of this AEI with the application context given by the user as well as the opening request user PDU.
 - On reception of the response, the other AEI attributes can be filled:
 - <Remote AEI access point>: takes the value contained in the PDU.
 - <Application context> takes the value contained in the response which can correspond to a reduction of that requested.
 - Passage of the AEI state to ESTABLISHED,
- The procedure which takes place at the responder AE level on reception of an opening request transmitted by the source procedure is the following:
 - Creation of an AEI object in the ASSOCIATED mode in the OPENING state.
 - Filling of the attributes of this object:
 - <Application Context>: takes the value indicated in the PDU, or a reduction, if the AE does not support it.
 - <Local AE address> takes the value received in the PDU.
 - <Local AEI access point> can take the optional value contained in the PDU or that returned by the RDF (Responder Directory Function).
 - <Remote AE address; Remote access point> take the values received in the PDU.
 - Addressing of the API from its implicit or explicit identification in the PDU.
 - Passing to the user of the opening request PDU.
 - When the API gives its response, a call of the A_ASSOCIATE.Rp service of MCSE in the framework of this AEI, with the PDU user, as a response to the opening request and possible updating of the context if this has been reduced by the API.
 - Passage of the AEI state to ESTABLISHED.

13.1.8.3 Exchange of information in the associated mode

When an API wishes to initialize an associated mode exchange, it proceeds as follows.

Exchange scenario:

- It addresses the AEI locally in the ESTABLISHED state, identified by its <AEI identifier> relative to its area by indicating:
 - the <application service> which must be executed with the called entity.
- The local procedure which must be performed at the AEI level is as follows:
 - During reception of the response, for a confirmed service, sending of this to the user.
 - Call of the data transfer service in the MCSE associated mode in the AEI framework, with as user data the PDU associated with the service.
- The procedure executed at the responder AEI on reception of the service request is as follows:
 - Passing of the <Application service> contained in the API PDU.
 - Service call and information transfer in the MCSE associated mode, in the framework of the AEI, with as user data the response PDU associated with this service.

13.1.8.4 Request for termination of an association.

When an API wishes to initialize a request for termination of an association, it proceeds as follows:

Closing scenario:

- It locally addresses the AEI identified by its <AEI identification> in its area, indicating:
 - the <Closing application service>.
- The local procedure which is executed at the AEI initiating this closing is as follows:
 - Setting of the AEI to the CLOSING state (Blocking of the data transfers beyond what has been engaged).
 - MCSE A-RELEASE service call to transmit the closing request service in the framework of this AEI.
 - On reception of the positive response, revocation of the AEI object, return to the ESTABLISHED state in the other cases;
- The remote procedure which is performed at the responder AEI level is as follows:
 - Setting of the AEI to the CLOSING state.
(Blocking of the data transfers outside those engaged).
 - Passing of the closing request service to the user.
 - At a positive response from the user, revocation of the AEI object and sending of the response to the initiator using the MCSE.

13.1.9 Concepts for Sub_MMS

13.1.9.1 General

Sub-MMS communications can be established through negotiated or predefined association as well as without association.

The assignment of access rights to association as well as the protections of objects, allow and the protection of objects authorizes selective access to the objects.

13.1.9.2 Sub-MMS communication channels

1- A Sub-MMS communication can be achieved without association:

These communication can take place in point to point (confirmed/unconfirmed services), in multipoint or in distribution (unconfirmed services).

In this case a service request (other than Initiate, Conclude and Abort) must be refused if the service is not supported, if the resources are insufficient, if it is not in compliance with the protocol, etc.

The Initiate, Conclude and Abort services have no significance.

This type of communication does not support the access protection mechanism on the objects.

2- A Sub-MMS communication can be performed in a predefined association:

These communication can take place in point to point (confirmed/unconfirmed services), in multipoint or in distribution (unconfirmed services).

In this case a service request (other than Initiate, Conclude and Abort) must be refused if the service is not supported, if it is not in compliance with the protocol or if the resources reserved for the association are insufficient, etc.

The Initiate, Conclude and Abort services have no significance.

This type of communication does/does not support the access protection mechanism on the objects.

3- A Sub-MMS communication can be performed in a negotiated association:

These communication can take place in point to point (services confirmed/unconfirmed),

In this case a service request (other than Initiate, Conclude and Abort) must be refused if the service is not supported, if it is not in compliance with the protocol, or if the resources reserved for the association are insufficient, etc.

This type of communication does/does not support the access protection mechanism on the objects.

13.1.9.3 Access protection mechanism

The access to Sub-MMS object can be refused to some clients. For this we define the protection levels assigned to an object and access right assigned to the association.

13.1.9.3.1 Protection level assigned to an object

The definition of objects other than the VMD object includes an OPTIONAL attribute called "ACCESS PROTECTION" which determines if an object is protected or not.

An object which is not protected has its "ACCESS PROTECTION" attribute forced to the value "FALSE".

An object which is protected has its "ACCESS PROTECTION" attribute forced to the value "TRUE".

- 1- Password,
- 2- Access Groups,
- 3- Access Rights.

Password:

The "Password" attribute allows restricting access only to clients holding the password.

Access Groups:

"Access Groups" attribute allows to restrict the access only to clients who belong to one of the groups defined by the attribute.

Access Rights:

The "Access Rights" attribute indicates the various possibilities to obtain access to the object.

It also indicates for each of these access possibilities, the services which can be executed as well as the conditions to be fulfilled.

Object class	Executable operations
Domain	Load, Upload, Delete, Connect to a program invocation
Program invocation	Execute, Initialize, Stop, Delete
Variable	Read, Write
Variable-List	Read, Write, Delete
Event	Read, Alter, Acknowledge
Access granted because of password	Conditions to be fulfilled
Yes	a) The association password is identical to that of the object.
No	b) The value of the password of the association is ZERO; condition (a) is not fulfilled
Access granted because of password	Conditions to be fulfilled
Yes	a) The group word for the association and the object have at least one identical group No
No	b) The value of the association group word is ZERO; condition (c) is not fulfilled
Access granted to all	Condition to be fulfilled
Yes	e) The "Access Rights" attribute authorizes access to all clients.

13.1.9.3.2 Level of the access rights assigned to an association

It is possible to assign access rights to predefined or negotiated associations. In this case these access rights are characterized by the parameters "Password" and "Access Groups".

The "Password" and "Access Group" parameters are granted implicitly to the predefined association.

The "Password" and "Access Group" parameters are granted to the negotiated association by the initiate service.

The "Password" and "Access Group" parameters thus characterize the Client's access rights.

13.1.9.3.3 Access control mechanisms for protected objects

Access to the objects can be granted:

- a) either when the "Password" attribute of the object is identical to the association "Password" parameter
- b) when the "Access Groups" attribute of the object and the "Access Groups" parameter of the association have at least one group No. in common.
- c) or to all the clients when the "Access Rights" attribute permits.

NOTE No access restriction is imposed for the use of the following services:

- Get Name List,
- Get Domain Attributes,
- Get Program Invocation Attributes,
- Get Variable Access Attributes,
- Get Variable List Attributes,
- Get Event Condition Attributes,
- Get OD Header/Data Type Attributes.

13.1.9.4 Directory management

13.1.9.4.1 General

All the sub-MMS object descriptors supported by an application form a directory.

The arrangement of the descriptors in a directory can be either standardized or free.

The standardization of the directory is performed by an object called "OD" (Object Dictionary).

13.1.9.4.2 Use of the "OD" object

It is not mandatory for a device to support the "OD" object.

13.2 Type 7 ASEs

13.2.1 MPS ASE (Periodic/Aperiodic Manufacturing Services)

13.2.1.1 Overview

In a MPS environment, a certain number of application objects are implemented. These objects come directly from the instantiation of the different classes previously described.

All these application objects representing the network configuration must be described partially or globally in order to allow:

- the user to have access to the application configuration (e.g.: Access to the variable type),
- the checking of the configuration consistency of the application between several subscribers (e.g.: configuration consistency between the producer and the consumers of a variable).

Moreover, this description is specified in this standard in order to ensure the interoperability in the configuration interchanges between different implementations.

13.2.1.2 Object concepts

13.2.1.2.1 Application objects

The description of periodic/apperiodic MPS services leads to describe all the resources involved in the application layer. This specification defines the static part of the specification.

The specification reactive aspects describe the various service elements actions on the application layer resources:

MPS environment application objects are divided into two sets:

- Objects related to variables,
- Objects related to variable lists.

Variables are abstract elements that associate a transfer syntax and a semantics to a data.

Variables, and related information, are provided to the user by the communication system, either individually, or by lists.

Other variables are provided with an event characteristic to which the user may associate actions relative to the execution of the distributed application..

13.2.1.2.2 Application objects addressing

The key-attributes (A_Name) used to address the MPS application layer objects all belong to the same addressing space.

Each of these names in the addressing space is defined within a maximum length of 16 characters.

The characters belong to the set defined for the "visible string" type of the abstract syntax notation ASN1 (ISO 8824).

Within that set, the following characters are to be used:

Upper case letters	(A..Z)
Lower case letters	(a..z)
Digits	(0..9)
Underscore	(<u>)</u>
Currency symbol	(\$)

The "space" character is not to be used.

Upper and lower case letters are considered as different characters.

An A_Name must not begin with a digit.

The "underscore" character as a second character defines a standard-reserved name.

13.2.1.3 Variable access Sub-ASE

13.2.1.3.1 Concepts

13.2.1.3.1.1 MPS environment variables

An MPS environment variable is an abstraction in the MPS environment, in which it is referenced by a unique name or description. The data associated with this variable corresponds to the representation handled by the data link services, which is exchanged on the network.

13.2.1.3.1.2 Network memory

Control system devices provide an AP with a value of each variable concerned an image of which is stored in local network memory.

13.2.1.3.1.3 Producer/Consumer/Third Party Concepts

Application entities that recognise a variable are provided with a local image of it.

Within a distributed application, one unique application entity is declared as the producer of the variable value whereas one or several application entities are declared as consumers of that value.

Moreover, some application entities, while being neither producer, nor consumer of the variable value, may have a knowledge of it in order to invoke the updating of its value. Such application entities are called third party entities.

13.2.1.3.1.4 Periodicity and aperiodicity concepts

The exchange of variable value between the producing entity and the consuming entities can be performed periodically or aperiodically.

A periodic updating is initiated by the network and is defined during the configuration step.

The updating period is determined by:

- The user requirements to fulfil the distributed application ,
- constraints related to the network.

Aperiodic updating is initiated by a user, such as the producing entity, a consuming entity, or a third party entity.

13.2.1.3.1.5 Synchronous and asynchronous properties

Application entities handling variables elaborated according to production validity, for the produced variables, and transmission validity for the consumed variables.

Definitions:

Production or transmission validity associated with each variable have an asynchronous character as they are elaborated independently from the network activity.

Production or transmission validity associated with each variable have a synchronous character when they involve synchronization orders issued from the network.

A synchronization order corresponds to the event associated with the emission or the reception of a variable of class SYNCHRONIZATION.

13.2.1.3.2 Variable Class specification

(see figure 73)

A variable in the MPS environment is modelled using the following objects:

- one *Produced Variable Local Image* object that includes all the attributes necessary to a producer user.
- one or several *Consumed Variable Local Image* object that includes all the attributes necessary to one or several Consumer users.

Those two object types are issued from inheritance from several classes:

The *Produced Variable Local Image* objects are instantiated from the *Produced Variable* class.

The *Produced Variable Class* includes all the attributes proper to a producing entity, and inherits from the *Identified Variable* class. That latter class includes all the attributes that are common to every local image associated with that variable.

Moreover, the *Produced Variable* class optionally inherits from two classes *Refreshment* and *Punctual Refreshment* that include all the attributes necessary to the elaboration of informations relative to the production validity associated with that variable, within the producing application entity.

The *Consumed Variable Local Image* objects are instantiated from the *Consumed Variable* class.

The *Consumed Variable* class includes all the variable attributes proper to a consuming entity, and inherits from the *Identified Variable* class. That latter class includes all the attributes common to every local image associated with that variable.

Moreover, the *Consumed Variable* class optionally inherits from two classes *Promptness* and *Punctual Promptness* that includes the attributes necessary to the elaboration of the informations relative to the transmission validity associated with the variable within the consuming application entities.

Each of the classes *Produced Variable* and *Consumed Variable* optionally inherits respectively form a class *Production Resynchronization* and *Consumption Resynchronization* that include the attributes relative to the double memorisation of a variable in order to allow the providing of the variable:

- to the network by a producing entity,
- To the consuming entities by the network, upon a synchronization order issued from the network.

All the common attributes of a variable are described in a *Variable* generic class.

The variable type is described in a *Type Constructor* class which is referred by the variable and that allow the elaboration of all the type that may exist in the MPS environment.

Concerning the access to a variable, the *Variable Access* class refers to a variable and indicates the access mode that is used.

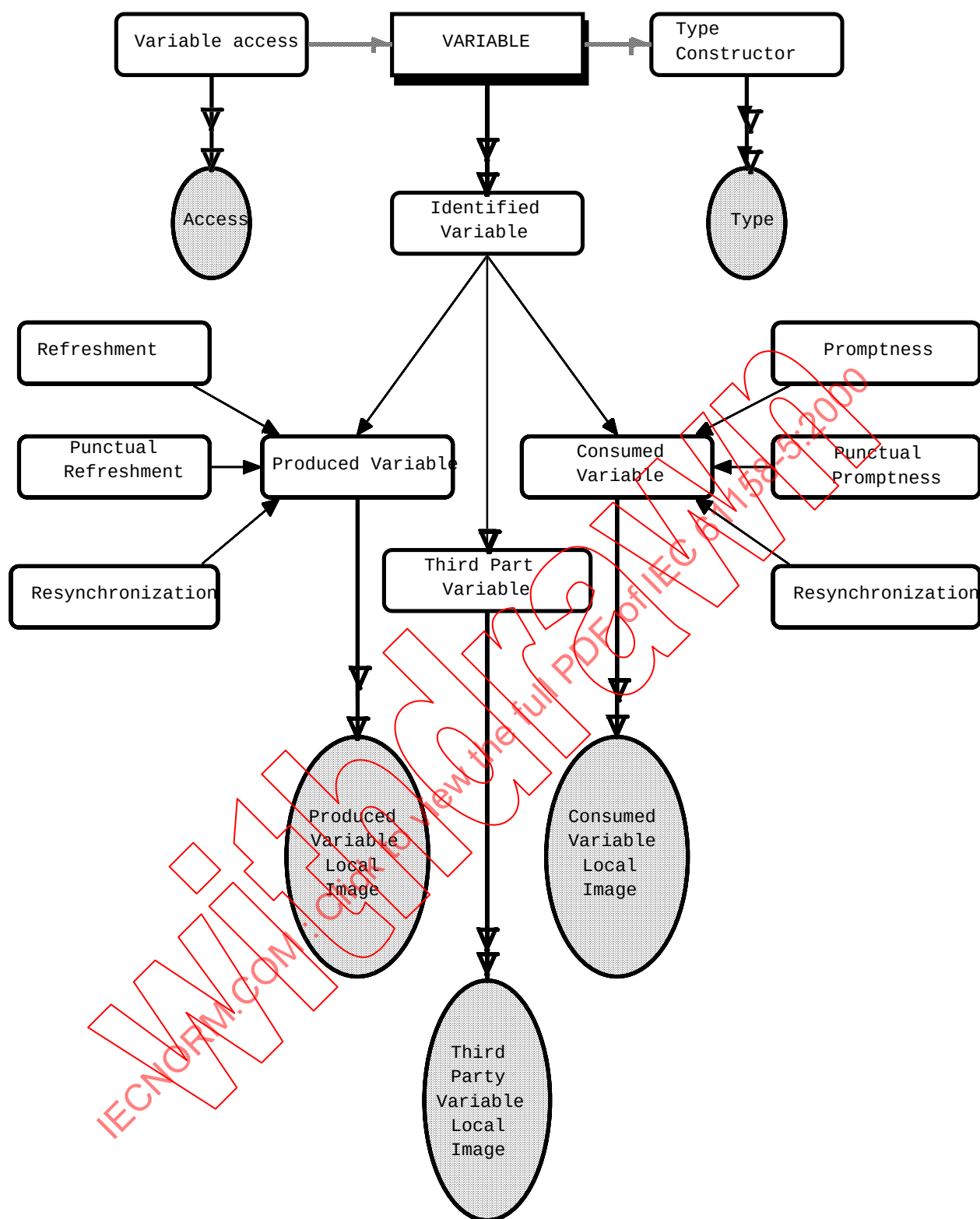


Figure 73 – Object model of the MPS ASE

13.2.1.3.2.1 Variable generic class specification**FAL ASE:** **MPS ASE****CLASS:** Variable**CLASS ID:** not used**PARENT CLASS:** not used

Generic Class: VARIABLE

Class Attributes:

Key-Attribute	:	A_Name
Attribute	:	Reference Type Constructor
Attribute	:	Transmitted status [TRUE, FALSE]
Constraint	:	Transmitted status = TRUE
Attribute	:	Significant Status
Attribute	:	Identifier
Attribute	:	Transmission mode [PERIODIC,APERIODIC]
Constraint	:	Transmission mode = PERIODIC
Attribute	:	Network period
Attribute	:	Class [NORMAL,SYNCHRONIZATION,DESCRIPTION]
Constraint	:	Class = SYNCHRONIZATION
Attribute	:	Consistency variable [TRUE, FALSE]
Constraint	:	Consistency variable = TRUE
Attribute	:	Reference Variable List

Instance Attributes:

Attribute	:	Public value
Constraint	:	Transmitted status = TRUE
Attribute	:	Transmitted status value
Attribute	:	Universal services requested [TRUE, FALSE]
Constraint	:	Universal services requested = TRUE
Attribute	:	Universal services scope [LOCAL,DISTANT]
Constraint	:	Universal services scope = DISTANT
Attribute	:	Priority [NORMAL,URGENT]

13.2.1.3.2.1.1 Attributes**A_Name:**

This attribute uniquely identifies, a variable at the interface between the application layer and the user. The A_Name of a variable belongs to the MPS addressing space.

Moreover, for a variable for which the Class attribute has the value NORMAL, or SYNCHRONIZATION the A_Name length is limited to 14 characters.

Reference Type Constructor:

This reference indicates the A_Name of a global type associated with the variable.

The type is described by a *Type Constructor* object that supports any variable type that may exist in MPS environment.

Transmitted status:

When this boolean attribute has the value TRUE, status relative to the variable production validity are broadcast to the respective consumers.

Significant status:

For the variables with information relative to their production validity, it is necessary to specify which status are effectively elaborated by the producer.

This attribute is of type Boolean array, each TRUE element indicates, that the production validity status of the associated the variable is significant.

The associated production validity status is defined for the *Transmitted status Value* array element with the same index

The various elements are:

S1: Elaboration by the producer of a synchronous or asynchronous refreshment status.

S2: Elaboration by the producer of a punctual refreshment status.

Identifier:

This attribute corresponds to a data link address that refers a data link layer object.

There is a one to one correspondence between a variable A_Name and the associated identifier. Moreover, the relationship between an identifier and a variable A_Name is a local issue committed during the network configuration generation step.

Transmission mode:

This attribute defines the exchanges of the value of the variable it is associated with.

The exchanges may be PERIODIC or APERIODIC, depending on the user requirements, and are defined during the network configuration step.

Network period:

When the transmission mode is PERIODIC, this attribute specifies the updating period of the variable by the network.

This period is expressed in milliseconds.

Class:

This attribute permits to associate a functionality to a variable. It may have one of the following values:

NORMAL: NORMAL class variables indicate process representation information.

SYNCHRONIZATION: SYNCHRONIZATION class variable indicates:

- Synchronization information of the distributed application,
- information necessary to the spatial consistency mechanism for a variable list.

DESCRIPTION: DESCRIPTION class variables features the description of all or part of the attributes relative to a variable, to a variable access, to a variable type, or to a variable list.

Consistency variable:

When this attribute has the value TRUE , the described variable corresponds to a consistency variable involved with the spatial consistency mechanism of a variable list.

Reference Variable list:

This reference to the *Variable List* generic class indicates, for a CONSISTENCY class variable, the variables the reception of which, within an application entity, takes part in the elaboration of the consistency variable value when a spatial consistency mechanism is involved

Public value:

This attribute corresponds to the value of the variable provided to the consumers by the producer.

After instantiation, that attributes includes:

- The public value provided to the network
- the last public value received from the network-

regarding that the instance corresponds to a producer's one or a consumer's one.

At a given time, it is to be considered the value available at the consumer, the value transmitted during the last exchange, and the last value received by each consumer.

The last value received by a consumer is not always identical to the last transmitted value because of possible transmission faults.

Transmitted status value:

This attribute corresponds to the array of production validity status , relative to a variable, and transmitted with that variable value. Each status within the array is a boolean.

All the production validity status associated with the variable value are not necessarily significant.

As for the variable public value, that attribute instantiation includes:

- The status values provided to the network,
- the last status values received from the network

regarding that the instance corresponds to a producer's one or a consumer's one.

The handling at a producer and at the consumers of the set consisting of the variable value and its associated status values is atomic.

Status array elements:

- S1 Synchronous/asynchronous refreshment status
- S2 Punctual refreshment status

NOTE The abstract syntax defines the array of status.

Requested universal services:

When this boolean attribute has the value TRUE , universal services may be invoked for the variable. in consideration.

Universal services scope:

This attribute is used by the variable access extended services, and defines the service category - local or distant- to use for the universal read/write services.

When that attribute has the value LOCAL , universal services that are used correspond to local services; when it has the value DISTANT , they correspond to distant services.

Priority:

This attribute is used by the variable access extended services and defines the priority level used to request variable updating when the universal services correspond to distant services.

When that attribute has the URGENT value, flow control guarantees that an urgent updating request will be processed before a normal request initiated after it.

13.2.1.3.2.2 Identified Variable class specification

The *Identified Variable* class is issued from the instantiation of the *Variable* generic class.

13.2.1.3.2.3 Produced Variable class specification

This class gathers all the attributes relative to a variable within a producer application layer.

FAL ASE:**MPS ASE****CLASS:**

Produced variable

CLASS ID:

not used

PARENT CLASS:

not used

Class: PRODUCED VARIABLE

Attribute	: <i>Inherit from</i> Identified Variable
Attribute	: Refreshment elaborated [TRUE, FALSE]
Constraint	: Refreshment elaborated = TRUE
Attribute	: <i>Inherit from</i> Refreshment
Attribute	: Punctual refreshment elaborated [TRUE, FALSE]
Constraint	: Punctual refreshment elaborated = TRUE,
Attribute	: <i>Inherit from</i> Punctual Refreshment
Attribute	: Resynchronized Variable [TRUE,FALSE]
Constraint	: Resynchronized Variable = TRUE
Attribute	: <i>Inherit from</i> Production Resynchronization
Attribute	: Emission Indication[TRUE,FALSE]

13.2.1.3.2.3.1 Attributes***Inherit from Identified Variable:***

Produced Variable class inherits from *Identified Variable* class that provides it with the attributes common to all local copies of the same variable within MPS environment.

Refreshment elaborated:

Refreshment is an information relative to a variable production validity.

This information, which is optionally elaborated, tells the consumer users about the production period of the producer user.

The status associated with this information is elaborated within the producer application layer and provided to the network in the status value associated with the variable value.

When this Boolean attribute has the value TRUE , a refreshment status is elaborated within the producer application entity.

Inherit from Refreshment:

The *Produced Variable* class inherits from the *Refreshment* class.

The *Refreshment* class includes all the attributes relatives to the elaboration of the refreshment information associated with a variable, in order to qualify its production.

Punctual Refreshment elaborated:

Punctual refreshment is an information relative to a variable production validity.

This information, that is optionally elaborated, tells consumer users about the respect by the producer user of a writing operation within a time slot associated with a synchronization order issued from the network.

The status associated with that information is elaborated within the producer application layer and provided to the network in the status value associated with the variable value.

When this boolean attribute has the value TRUE , a punctual refreshment status is elaborated within the producer application entity.

Inherit from Punctual Refreshment:

The *Produced Variable* class inherits from the *Punctual Refreshment* class.

The *Punctual Refreshment* class includes all the attributes relative to the elaboration of the punctual refreshment information associated with the variable, in order to qualify its production.

Resynchronized Variable:

Some users produce variables asynchronously with respect to the network.

These variables can be resynchronized within the application layer by the means of a double buffer.

The private buffer is the one provided to the asynchronous producer user. The public buffer is the one provided to the network.

Resynchronization of such a variable consists of transferring from Private Buffer to Public Buffer upon the reception of a synchronization order from the network.

This attribute supports the chosen option for the variable concerned.

When this boolean attribute has the value TRUE , the concerned variable is resynchronized in production.

Inherit from Production Resynchronization:

The *Produced Variable* class inherits from the *Production Resynchronization* class.

The *Production Resynchronization* class includes all the attributes relative to a variable resynchronization.

Emission Indication:

When it has the value TRUE this boolean attribute indicates that an emission indication (A_SENT.ind) has to be returned to the user once the variable has been sent on the network.

13.2.1.3.2.4 Produced Variable Local Image class specification

Produced Variable Local Image object is issued from the instantiation of the *Produced Variable* Class.

13.2.1.3.2.5 Consumed Variable class specification

This class gathers all the attributes relative to a variable within a consumer application layer.

FAL ASE:	MPS ASE
CLASS:	Consumed variable
CLASS ID:	not used
PARENT CLASS:	not used

Class: CONSUMED VARIABLE

Attribute	: Inherit from Identified Variable
Attribute	: Promptness elaborated [TRUE, FALSE]
Constraint	: Promptness elaborated = TRUE
Attribute: Inherit from Promptness	
Attribute	: Punctual Promptness elaborated [TRUE, FALSE]
Constraint	: Punctual Promptness elaborated = TRUE
Attribute: Inherit from Punctual Promptness	
Attribute	: Resynchronized Variable [TRUE, FALSE]
Constraint	: Resynchronized Variable = TRUE
Attribute: Inherit from Consumption resynchronization	
Attribute	: Reception Indication [TRUE, FALSE]

13.2.1.3.2.5.1 Attributes***Inherit from Identified Variable:***

Consumed Variable class inherits from *Identified Class* that provides it with all the attributes common to every local copy of the same Identified Variable within MPS environment.

Promptness elaborated:

Promptness is an information relative to a variable transmission validity.

This information, that is optionally elaborated, tells a consumer user about the reception of a variable respecting a consumption period

The status associated with this information is elaborated within the consumer application layer and provided to the consumer user.

When this boolean attribute has the value TRUE , a promptness status is elaborated within the consuming application entity.

Inherit from Promptness:

Consumed Variable class inherits from Promptness class.

Promptness class includes all the attributes relative to the elaboration of a promptness status associated with a consumed variable.

Punctual Promptness elaborated:

Punctual promptness is an information relative to a variable transmission validity.

This information, that is optionally elaborated, tells a consumer user about the reception of a variable within a time slot associated with a synchronization order issued from the network.

The status associated with this information is elaborated in the application layer and provided to the consumer user.

When this boolean attribute has the value TRUE , a punctual promptness status is elaborated within the consuming application entity.

Inherit from Punctual Promptness:

Consumed Variable class inherits from the Punctual Promptness class.

Punctual Promptness class includes all the attributes relative to the elaboration at the consumer level of a punctual promptness status associated with a variable.

Resynchronized Variable:

Some users consume variables asynchronously with respect to the network.

These variables can be resynchronized within the application layer by the means of a double buffer.

The private buffer is the one provided to the asynchronous consuming user. The public buffer is the one receiving the value from the network.

Resynchronization of such a variable consists in transferring the public buffer into the private buffer upon a synchronization order from the network.

This attribute supports the chosen option for the variable concerned in consumption resynchronization.

When this attribute has the value TRUE , the associated variable is resynchronized in consumption.

Inherit from Consumption Resynchronization:

Consumed Variable class inherits from the Consumption resynchronization.

Consumption Resynchronization includes all the attributes relative to a variable resynchronization in consumption.

Reception Indication:

When it has the value TRUE this boolean attribute indicates that a reception_indication (A_RECEIVED.ind) has to be returned to the user once the variable has been received from the network.

13.2.1.3.2.6 Consumed Variable Local Image class specification

Consumed Variable Local Image object is issued from the instantiation of *Consumed Variable* class.

13.2.1.3.2.7 Third Party Variable class specification**FAL ASE:** **MPS ASE****CLASS:** Third party variable**CLASS ID:** not used**PARENT CLASS:** not used

Class: THIRD PARTY VARIABLE

Attribute : Inherit from IdentifiedVariable

13.2.1.3.2.7.1 Attributes*Inherit from IdentifiedVariable*

Third Party Variable class inherits from *Identified Variable* class this provides it with the attributes necessary to know the variable.

Consequently, this variable is only provided with the pair (A_Name, Identifier) of the identified variable it inherits from.

13.2.1.3.2.8 Third Party Variable Local Image object class specification

Third Party Variable Local Image object is issued from instantiation of Third Party Variable class.

13.2.1.3.2.9 Type Constructor class specification**FAL ASE:** **MPS ASE****CLASS:** Type constructor**CLASS ID:** not used**PARENT CLASS:** not used

Class : TYPE CONSTRUCTOR

Key-Attribute : A_Name

Attribute : Construction

[SIMPLE, ARRAY, STRUCTURE, PREDEFINED, EXPLICIT]

Constraint : Construction = SIMPLE

Attribute : Primitive type

Attribute : Size

Constraint : Construction = ARRAY

Attribute : Compressed [TRUE, FALSE]

Attribute : Dimension

Attribute : Reference Type Constructor

Constraint : Construction = STRUCTURE

Attribute : List of

Attribute : Named field [TRUE, FALSE]

Constraint : Named field = TRUE

Attribute : field name

Attribute : Reference Type Constructor

13.2.1.3.2.9.1 Attributes

A-Name:

This attribute uniquely identifies an object within the application layer. An_A_Name belongs to the MPS addressing space. Moreover, a Type Constructor A_Name must not be longer than 14 characters.

An A_Name beginning with the characters K_ indicates a type reserved by the standard.

Construction:

The specified type can be of SIMPLE construction, and is then directly derived from the primitive types.

The specified type can be of ARRAY construction, and represents an ordered set of elements of the same type; that latter type can be itself of construction SIMPLE, ARRAY, or STRUCTURE. ARRAY construction type elements are increasingly numbered from zero (0) for the first element.

The specified type can be of STRUCTURE construction, and represents a complex type composed of an ordered set of one or several components. These components may have different types of SIMPLE, ARRAY, or STRUCTURE construction..

The specified type can be of PREDEFINED construction, and is then defined within the standard.

PREDEFINED construction types cannot be used to compose types of ARRAY or STRUCTURE .

The specified type can be of EXPLICIT construction in order to characterise a variable the type of which can change dynamically.

EXPLICIT construction types cannot be used to compose types of ARRAY or STRUCTURE construction.

SIMPLE construction types characteristics

Primitive Type:

Primitive types used by AL TYPE 7.

Size:

This attribute has a value and purpose that depends on the value of the Primitive Type attribute defined above. The size, described for each of the primitive types , for Type Constructor class instances, describes a maximum value to the variable.

ARRAY construction types characteristics

Compressed:

The application layer transfer syntax allows to define, for some ARRAY type variables, compressed encoding rules.

Compressed encoding is specified using a boolean attribute.

When this attribute has the value TRUE , the associated variable value encoding is compressed, each time it is possible depending on the array elements type; when this attribute has the Value FALSE, the associated variable value encoding is not compressed.

Dimension:

This attribute defines the number of elements within the ARRAY construction type.

Reference Type Constructor:

An ARRAY construction type is composed of elements of any one type either of construction SIMPLE, or ARRAY, or STRUCTURE.

This attribute references the "Type" object, instantiated from the *Type Constructor* class associated with the array elements

STRUCTURE construction types characteristics**Named field:**

When a type is STRUCTURE, each component composing the type corresponds to the semantic representation of the associated variable. In order to permit a partial access on these components they may be designated with a name. When this attribute has the value TRUE, a name is associated with the corresponding component; when it has the value FALSE, no name is associated with the component.

Field name:

This attribute is a character string that uniquely identifies a structured variable component.

NOTE The scope of component names is local to a structure.

Reference Type Constructor:

A STRUCTURE construction type is composed of components of any type. i.e. SIMPLE, ARRAY, or STRUCTURE.

This attribute denotes that the instance of *Type Constructor* class associated with each component is of type STRUCTURE

PREDEFINED construction types characteristics

PREDEFINED construction types are mostly used for description variables and refer to types that are reserved by the standard.

PREDEFINED construction types cannot be explicitly defined because they need a grammatical extension in order to use optional types, alternative types, or variable length types.

NOTE Grammatical extensions that are involved are specified in ISO 8824 standard.

EXPLICIT construction types characteristics

EXPLICIT construction types allow the associated variables to be provided with an explicit transfer syntax, in order to carry the type semantics with the data.

13.2.1.3.2.9.2 Primitive types

Primitive types that are selected, and their characteristics are mostly specified in the ISO 8824 standard.

NOTE The only access mode available for a primitive type variable is global access.

• BOOLEAN

The definition of this primitive type is the same as the boolean type definition in the ISO 8824 standard. For this type, the *Size* attribute must be omitted.

• BIT STRING

The definition of this primitive type is the same as the bit string definition in ISO 8824. For this type the *Size* attribute defines the number of bits in the bit string.

• INTEGER

The definition of this primitive type is the same as the integer type definition in ISO 8824 standard. For this type, the *Size* attribute defines the number of bits that are necessary to have a twos complement representation of all possible values.

• UNSIGNED

The definition of this primitive type is the same as the integer type definition in ISO 8824 with the exclusion of negative numbers. For this type, the *Size* attribute defines the number of bits that are necessary to have a twos complement representation of all possible values.

• OCTET STRING

The definition of this primitive type is the same as the octet string type definition in ISO 8824 standard. For this type, the *Size* attribute defines the number of octets in the string.

• VISIBLE STRING

The definition of this primitive type is the same as the visible string type definition in ISO 8824 standard. For this type, the *Size* attribute defines the number of characters in the string.

• GENERALISED TIME

This primitive type is defined with a fourteen-digits visible string (e.g. <<0 MDDHHMMSS>>); it is equivalent to a particular form of the generalised time type in ISO 8824 standard. For this type, the *Size* attribute must be omitted.

• FLOATING POINT

This primitive type defines value representation for positive and negative real numbers, including, zero, negative infinite and positive infinite.

The values at the limits (0, +∞, -∞), as well as the various representation formats are defined in ANSI/IEEE 754 standard.

For this type, the *Size* attribute, of boolean type, indicates whether the representation format is simple precision (FALSE) or double precision (TRUE).

• BINARY TIME

This primitive type is defined as the unsigned type of the present standard, the value of which corresponds to a number of elementary times.

For this type, each value of the *Size* attribute defines the value of an elementary time as well as a number of bits used to represent any possible binary time value:

Size	10 ms	0,1 ms	1 ms	10 ms
16 Bits	0	1	2	3
32 Bits	4	5	6	7
48 Bits	8	9	XXXXX	XXXXX

Table 311 – Binary time coding

• BCD

This type is used to represent a set of values corresponding to the numeric representation of digits.

For this type, the *Size attribute* must be omitted.

13.2.1.3.3 Basic Mechanisms

13.2.1.3.3.1 Time-out class specification

The elaboration, by producing application entities ,of information relative to production validity involves time-out resources. Likewise for consuming application entities.

The description of this resource can be modelled with an attribute object, whose purpose is to define visible aspects of time-out used by MPS environment services.

FAL ASE: **MPS ASE**

CLASS: Time-out

CLASS ID: not used

PARENT CLASS: not used

Class: TIME-OUT

Attribute : Preset

Attribute : Current value

Attribute : State [RUNNING, IDLE]

13.2.1.3.3.1.1 Attributes

Preset:

Preset corresponds to the duration associated with the time-out when it is armed, whenever the time-out is running or expired.

Current value:

Current value corresponds to the instantaneous value of the time-out.

That value is equal to:

- the period remaining before expiration, when the time-out is running
- "0" when the time-out has expired.

State:

The time-out state indicates whether the time-out is active or not.

The two states are as follows:

RUNNING:Corresponds to a running time-out within the time interval between preset value and expiration.

IDLE: Corresponds to an expired time-out.

NOTE As soon as the time-out current value is forced with a non-zero value, the time-out is automatically set to RUNNING state.

13.2.1.3.3.1.2 Time-out mechanism

The dynamic working of a time-out is defined by the following evaluation net:

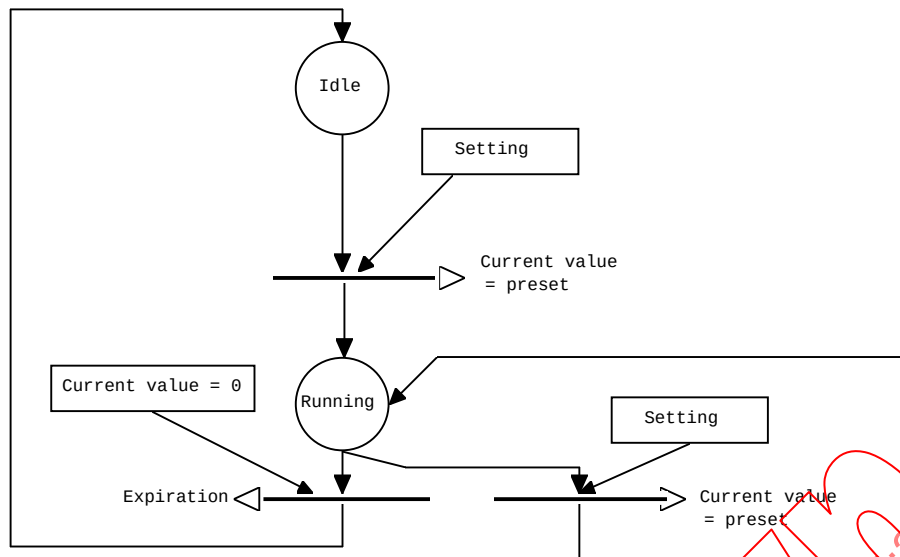


Figure 74 – Time-out evaluation net

13.2.1.3.3.2 Variable addressing

User access to application variables in MPS environment can be carried out in various ways.

Access to a variable is carried out by means of:

- a variable A_Name, in the case of variable global access,
- an access path in the case of explicit partial access to a structured variable named field or to an array element,
- an access A_Name, in the case of renamed partial access to a variable or to one of its components (Structure field, array element...)

Access A_Name belong all to the same MPS addressing space provided to the user at the interface with the application layer

13.2.1.3.3.2.1 Access modes

13.2.1.3.3.2.1.1 Global access

This access mode allows the user to globally access a variable.

The access to the variable is made with the Application name of the variable that is accessed. In this case, the application name corresponds to the A_Name attribute of *Produced Variable Local Image*, or *Consumed Variable Local Image*, or *Third Party Variable Local Image*. It belongs to the MPS addressing space provided to the user at the interface between the application layer and the user.

13.2.1.3.3.2.1.2 Explicit partial access

This access mode allows the user the access to a named field of a structured type variable or to an element of an array type variable.

The explicit partial access is made with an access path corresponding to the enumeration of all successive fields in the structure tree or array elements index; This enumeration must begin from the variable root, to the component that is accessed.

13.2.1.3.3.2.1.3 Renamed access

Renamed access consists in the projection of an A_Name from the MPS addressing space on a variable A_Name, or on a structured type variable access path, or on an array type variable element.

The names that are used in renamed partial access belong to the same addressing space as variable global names; this means they belong to the addressing space provided to the user at the interface with the application layer.

Consequently, renamed access allows the user access to a structure field or to an array element in the same way as for a global variable.

NOTE This access mode is provided to improve transmission efficiency, by grouping several user defined variables in the same structure .

This means that the transmission efficiency improvement can lead to group in a minimum number of frames variables exchanged between APs in an elementary distributed application. This is done in order to minimise control data in the data link layer.

Optimisation structures created at the application layer are transparent to the user with the renamed access, but they have to be declared in all the application entities that have to access all or part of the user defined variables they represent.

Note on optimisation structure elaboration:

Only variables which do not need production validity and for which transmission validity is elaborated with the same value of the consumption period can be grouped within the same optimisation structure.

13.2.1.3.3.2.2 Variable renamed access modelization

13.2.1.3.3.2.2.1 Variable Access class specification

Class: VARIABLE ACCESS

Key-Attribute	: A_Name
Attribute	: Reference Variable
Attribute	: Access Mode [PARTIAL, GLOBAL]
Constraint	: Access Mode = PARTIAL
Attribute	: Access Path

A_Name:

This attribute allows a unique specification of an access name at the interface between the application layer and the user. The A_Name belongs to the MPS addressing space

Moreover, a variable access A_Name must not be longer than 14 characters.

Reference Variable:

The *Variable Access* class refers to the *Variable* class.

Variable class indicates the application variable with which the access A_Name is associated.

NOTE A variable renamed access name is global, this means that the same access name within several different application entities must refer to the same application layer variable.

Access Mode:

This attribute indicates which access mode is to be used.

When it has the value TRUE, this attribute indicates a global renamed access to an application layer variable.

When it has the value FALSE, it indicates a partial access to a field of a structured variable or to an array element.

Access Path:

The access path to a structured variable field or to an array element refers, by increasing depth order structure field names or array element index.

NOTE In order to reach a full specification of the access path to a variable component, it is necessary that all the *Named Field* attributes of the *Type Constructor* objects at the various level of the structure have the value TRUE.

13.2.1.3.3.3 Variable transmission validity specification

Consumer users may be provided with information relative to transmission validity for a variable.

Two sets of transmission validity information are defined:

- Promptness; relative to transmission validity regarding a user validity delay for the variable that is received: this validity delay is called the consumption period. Semantics associated with this information depends on the synchronous or asynchronous nature of the promptness
- Punctual promptness; that is relative to the transmission validity regarding a time slot associated with a synchronization order issued from the network.

13.2.1.3.3.3.1 Promptness

Promptness attributes associated with variable are optional.

Promptness is relative to the validity of availability of a variable value provided to the user by the network. This validity is elaborated with a maximum consumption period which is fixed by the user.

In the case of synchronous promptness, the semantics of that validity information takes into account the respect of availability by the network of a synchronization order associated with the variable.

Asynchronous Promptness:

When it has the value TRUE, this boolean information indicates in a consumer entity that the variable value has been updated by the network for a delay that is smaller than the variable consumption period.

Synchronous Promptness:

When it has the value TRUE, this boolean information indicates in a consumer entity that the synchronization order has been received and has been followed by (at least) one updating of the associated variable by the network, and that the updating was within the consumption period.

13.2.1.3.3.3.1.1 Promptness class specification

This class gathers all the attributes relative to the elaboration of a promptness information associated with a variable within a consuming application entity.

Class: PROMPTNESS

Attribute	: Consumption period
Attribute	: <i>Inherit from</i> Time-out
Attribute	: Promptness characteristic [SYNCHRONOUS, ASYNCHRONOUS]
Constraint	: Promptness characteristic = SYNCHRONOUS
Attribute	: <i>Reference</i> (Synchronization) Variable
Attribute	: Promptness status [TRUE,FALSE]

Consumption period:

The consumer user declares a consumption period corresponding with the needs_of the distributed application.

This period corresponds to the minimal period to which the consumer user will access the variable.

Inherit from Time-out:

Promptness class inherits from Time-out class.

Time-out class includes all the attributes relative to a time-out. This resource is mandatory to elaborate an information relative to a variable promptness.

Promptness characteristics:

Promptness may have a characteristic which is proper to the consumer, and that influences the promptness status semantics.

When the characteristic is ASYNCHRONOUS, promptness is elaborated exclusively with attributes proper to the variable.

When the characteristic is SYNCHRONOUS, promptness takes into account not only the variable attributes, but also an event related to the reception of a synchronization order.

Reference (Synchronization) Variable:

This attribute references a *Variable* class for which the *Class* attribute has the value SYNCHRONIZATION. This allows reference to the synchronization order which sets the time-out in the case of synchronous promptness.

Promptness status:

Variable promptness is characterized by a status.

This status is of type boolean and is locally elaborated within each entity having a *Consumed Variable Local Image* associated with the variable.

Promptness status transition conditions from one state to the other are elaborated by the means of:

- Events corresponding to the variable reception or to the occurrence of the synchronization order associated with the variable,
- The expiration of the time-out associated with the variable,

- The state of the time-out associated with the variable for the processing of synchronous promptness.

13.2.1.3.3.1.2 Mechanism

The asynchronous promptness status associated with a variable within a consuming entity is dynamically processed by the following mechanism:

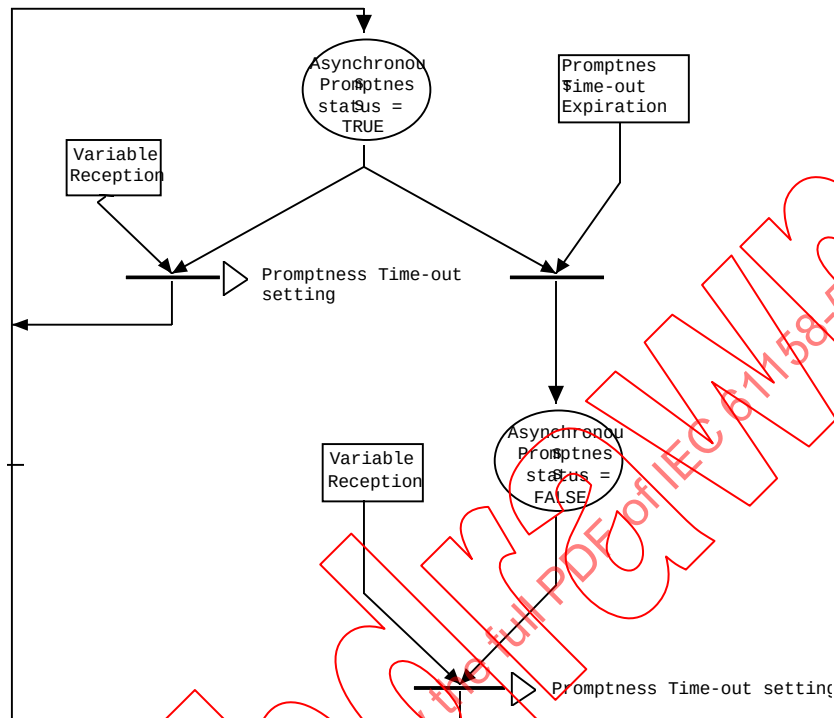


Figure 75 – Asynchronous promptness status evaluation net

Initial state:	<u>Asynchronous Promptness status = FALSE</u>
	Promptness time-out state = IDLE
Requests:	<u>Variable Reception:</u>
	Variable available from the network
	<u>Promptness time-out expiration:</u>
	Transition of the time-out associated with the mechanism elaborating the asynchronous promptness status from RUNNING state to IDLE state.
Actions:	<u>Promptness time-out setting</u>
	Transition of the time-out associated with the mechanism elaborating the asynchronous promptness status from IDLE state to RUNNING state with a preset equal to the <i>Consumption period</i> attribute value.

Synchronous promptness status associated with a variable within a consuming application entity is processed with the following mechanism:

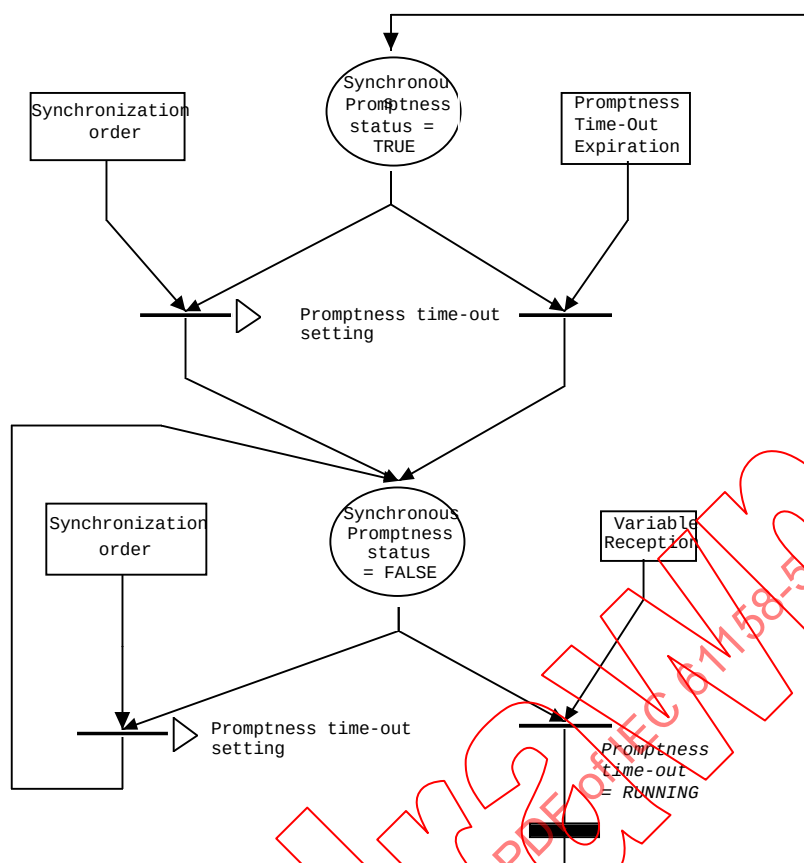


Figure 76 – Synchronous promptness status evaluation net

Initial state:	<u>Synchronous promptness status = FALSE</u>
	Promptness time-out state = IDLE
Requests:	<u>Variable reception:</u>
	Variable available from the network
	<u>Synchronization order:</u>
	Occurrence of a synchronization order associated with the synchronous promptness status elaboration mechanism.
	<u>Promptness time-out expiration:</u>
	Transition of the time-out associated with the mechanism elaborating the synchronous promptness status from RUNNING state to IDLE state.
Actions:	<u>Promptness time-out setting</u>
	Transition of the time-out associated with the mechanism elaborating the synchronous promptness status from IDLE state to RUNNING state with a preset equal to the <i>Consumption period</i> attribute value.

13.2.1.3.3.2 Punctual promptness

Punctual promptness information associated with a variable is optional.

Punctual promptness is relative to a validity of availability of the variable value to the consumer user from the network. That validity is elaborated by the means of a consumption time-slot that is

defined during the configuration step. This time-slot is activated upon reception of a synchronization order from the network.

Punctual Promptness:

When it has the value TRUE this boolean informs a consuming application entity that the variable value has been updated by the network (at least) once in the time slot following the reception of a synchronization order associated with the variable. It indicates also that the variable value has not been updated outside the time slot.

13.2.1.3.3.2.1 Punctual Promptness class specification

This class gathers all the attributes relative to the elaboration of a punctual promptness information associated with a variable within a consuming application entity.

Class: PUNCTUAL PROMPTNESS

Attribute	: Consumption time slot
Attribute	: Inherit from Time-out
Attribute	: Reference (Synchronization) Variable
Attribute	: Punctual Promptness Status [TRUE, FALSE]

Consumption time-slot:

Punctual promptness is elaborated by the means of a consumption time slot associated with a variable by a consumer user.

Punctual promptness informs a variable consumer that it has received a new value (at least) once during the time slot associated with the last synchronization order that has been received.

Inherit from Time-out:

Punctual Promptness class inherits from Time-out class.

Time-out class includes all the attributes relative to a time-out (preset, state).

Reference (Synchronization) Variable:

This attribute refers to a *Consumed Variable* class the *Class* attribute of which has the value SYNCHRONIZATION, and that defines the synchronization order that sets the time-out in the case of a punctual promptness.

Punctual promptness status:

A variable punctual promptness is characterized by a status.

That status is of type boolean and is locally elaborated within each entity where a *Consumed Variable Local Image* is declared for the variable.

Promptness status transition conditions from one state to the other are elaborated by the means of:

- Events corresponding to the reception of the variable or to the occurrence of a synchronization order associated with that variable,
- State of the time-out associated with the variable for the punctual promptness processing.

13.2.1.3.3.2.2 Mechanism

Punctual promptness status associated with a variable within a consuming application entity is processed with the following mechanism:

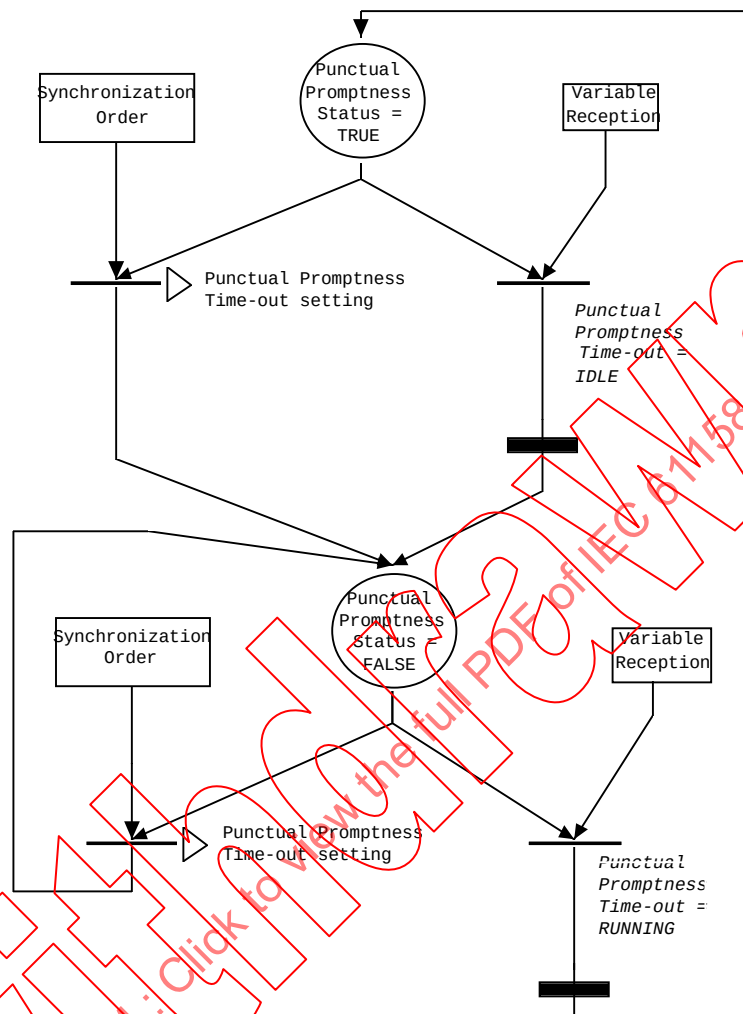


Figure 77 – Punctual promptness status evaluation net

Initial State:

Punctual Promptness status = FALSE

Promptness time-out state = IDLE

Requests:

Variable Reception:

Variable available from the network

Synchronization order:

Occurrence of the synchronization order associated with the punctual promptness elaboration mechanism.

Actions:

Punctual Promptness time-out setting

Transition of the time-out associated with the mechanism elaborating the punctual promptness status from IDLE state to RUNNING state with a preset equal to the *Consumption Time slot* attribute value.

13.2.1.3.3.4 Variable production validity specification

Production validity may be associated with any variable. Validity information is elaborated by a producing application entity and provided to the consuming entities. Those consuming entities only see a sampled value of the information.

Two categories of production validity information are defined:

- Refreshment, relative to a production validity regarding a maximum delay fixed by the user. Refreshment semantics depends on its characteristic, synchronous or asynchronous, within a producing application entity.
- Punctual refreshment relative to production validity regarding a production time slot associated with a synchronization order issued from the network.

13.2.1.3.3.4.1 Refreshment

This production validity information is elaborated as an option and is characterized with a status.

Refreshment is relative to validity of availability to the network by a producer user of a variable value. This validity is elaborated by the means of a production period that is fixed by the user.

In the case of a synchronous refreshment, the semantics of this validity information denotes the respect of availability by the network of the synchronization order associated with the variable.

That refreshment information, carried with the variable value by the network, is provided to consumer users.

Asynchronous refreshment:

When it has the value TRUE this boolean indicates in a producing application entity that the variable value has been provided to the network by the producer user, with a delay less than the production period.

Synchronous Refreshment:

When it has the value TRUE this boolean indicates within a producing application entity that a synchronization order has been received and followed by a (at least) one sourcing of a new variable value to the network from the producer user with a delay less than the production period.

13.2.1.3.3.4.1.1 Refreshment class specification

This class gathers all the attributes relative to the elaboration of a refreshment information associated with a variable within a producing application entity.

Class: REFRESHMENT

Attribute	: Production period
Attribute	: Inherit from Time-out
Attribute	: Refreshment characteristic [SYNCHRONOUS, ASYNCHRONOUS]
Constraint	: Refreshment characteristic = SYNCHRONOUS
Attribute	: Reference (Synchronization) Variable
Attribute	: Refreshment Status [TRUE, FALSE]

Production period:

The producer user may declare a production period corresponding to the needs of its distributed application.

This period corresponds to the maximum interval during which the producer user will provide the variable to the network.

Inherit from Time-out:

Refreshment class inherits from *Time-out* class.

Time-out class includes all the attributes relative to a time-out. This resource is mandatory to the elaboration of information relative to variable refreshment.

Refreshment characteristic:

Refreshment may be provided with a characteristic proper to a producer and that influences the refreshment status semantics.

When the characteristic is **ASYNCHRONOUS**, refreshment is elaborated exclusively with attributes proper to the variable.

When the characteristic is **SYNCHRONOUS**, refreshment takes into account not only the variable attributes, but an event related to the reception of a synchronization order.

Reference (Synchronization) Variable:

The attribute refers to a *Variable* class, the *Class* attribute of which has the value **SYNCHRONIZATION**. This defines the synchronization order which sets the time-out in the case of a synchronous refreshment.

Refreshment Status:

The variable refreshment provided to a user is characterized by a status.

This status is of type boolean and is locally elaborated within each application entity where a *Produced Variable Local Image* is declared.

The value of the status is associated with the variable value during the exchange on the bus.

Refreshment status is defined by two states, and the transition conditions from one state to the other are elaborated by means of:

- Events corresponding to the production of the variable,
- Events related to the occurrence of the synchronization order associated with the variable,
- Expiration of the time-out associated with the variable,
- State of the Time-out associated with the variable for the processing of synchronous refreshment.

13.2.1.3.3.4.1.2 Mechanism

Asynchronous refreshment status associated with a variable within a producing application entity is elaborated by means of the following mechanism:

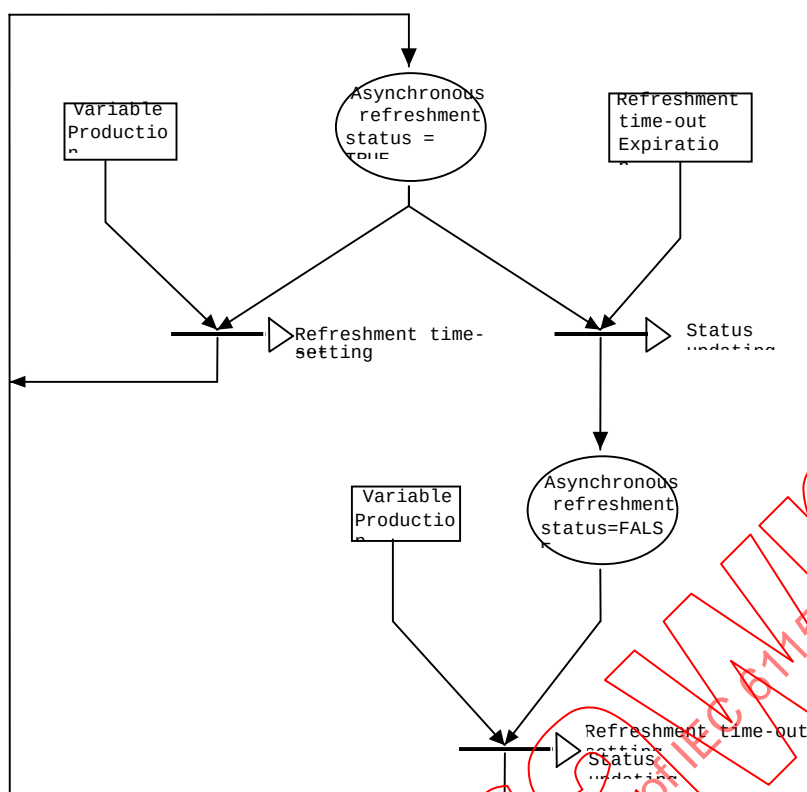


Figure 78 – Asynchronous refreshment status evaluation net

Initial state: Asynchronous refreshment status = FALSE
Refreshment time-out state = IDLE

Requests: Variable Production:
A_WRITELOC, A_WRITEFAR or A_WRITE service invocation by the user

Refreshment time out expiration:
Transition of the time-out associated with the asynchronous refreshment evaluation mechanism from the state *RUNNING* to the state *IDLE*

Action: Refreshment time-out setting:
Transition of the time-out associated with the asynchronous refreshment mechanism from the state *IDLE* to the state *RUNNING*, with a preset equal to the *Production Period* attribute value.

Status updating:
Providing of the asynchronous refreshment status value to the network.

Synchronous refreshment status associated with a variable within a producing application entity is elaborated by means of the following mechanism:

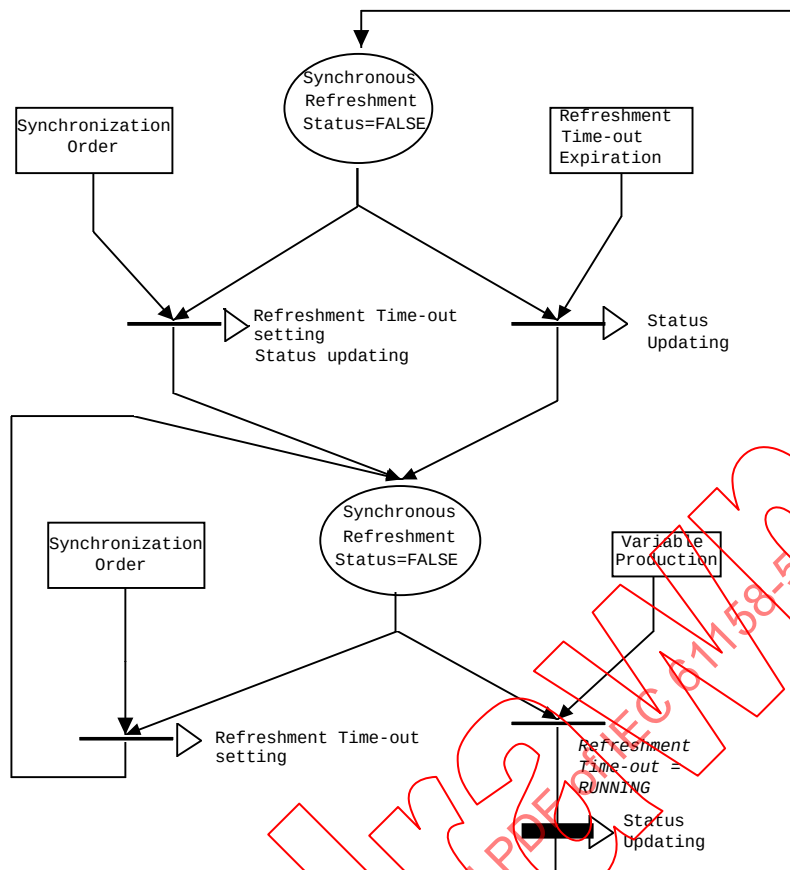


Figure 79 – Synchronous refreshment status evaluation net

Initial state: Synchronous refreshment status = FALSE

Refreshment time-out state = IDLE

Requests: Variable Production:

A_WRITELOC, A_WRITEFAR or A_WRITE service invocation by the user

Refreshment time out expiration:

Transition of the time-out associated with the synchronous refreshment evaluation mechanism from the state *RUNNING* to the state *IDLE*

Synchronization order:

Occurrence of the synchronization order associated with the synchronous refreshment elaboration mechanism.

Action:

Refreshment time-out setting:

Transition of the time-out associated with the synchronous refreshment mechanism from the state *IDLE* to the state *RUNNING*, with a preset equal to the *Production Period* attribute value.

Status updating:

Providing of the asynchronous refreshment status value to the network.

13.2.1.3.3.4.2 Punctual refreshment

Production validity information is optional.

Punctual refreshment is relative to the validity of availability to the network by the producer user of a variable value. This validity is elaborated by the means of a time-slot fixed by the user. This time slot is activated upon the reception of a synchronization order from the network.

Punctual Refreshment:

When it has the value TRUE this boolean indicates within a producing application entity that a variable value has been provided to the network by the producer user (at least) once during the time slot following the reception of a synchronization order. The variable value shall not have been provided outside the time slot.

Punctual refreshment status is associated with the variable value and is provided to the network.

13.2.1.3.3.4.2.1 Punctual Refreshment class specification

This class gathers all the attributes relative to the elaboration of a punctual refreshment information associated with a variable within a producing application entity.

Class: PUNCTUAL REFRESHMENT

<i>Attribute</i>	: Production time slot
<i>Attribute</i>	: <i>Inherit from</i> Time-out
<i>Attribute</i>	: <i>Reference</i> (Synchronization) Variable
<i>Attribute</i>	: Punctual Refreshment Status [TRUE,FALSE]

Production Time slot:

Punctual refreshment is elaborated by the means of a production time-slot associated with a variable by the producer user. Punctual refreshment allows a consumer user to check whether the variable producer guarantees its production within the time slot activated by the last synchronization order issued from the network, and associated with it.

Inherit from Time-out:

Punctual Refreshment class inherits from *time-out* class.

Time-out class includes all the attributes relative to a time-out (preset, state) that characterize the production time slot.

Reference (Synchronization) Variable:

This attribute refers to a *Variable* class, the attribute *Class* of which has the value SYNCHRONIZATION, and that permits to define the synchronization order that sets the time-out in the case of a punctual refreshment.

Punctual Refreshment Status:

Variable punctual refreshment is characterized by a status.

This status is of type boolean and is locally elaborated within each entity owning a *Produced Variable Local Image* object of the variable.

Punctual refreshment status is characterized by two states, the transition conditions between which are elaborated by the means of:

- Events corresponding to the variable production, or to the occurrence of a synchronization order associated with the variable,
- State of the time-out associated with the variable for the processing of the punctual refreshment.

13.2.1.3.3.4.2.2 Mechanism

Punctual refreshment status associated with a variable, within a producing application entity, is elaborated by the means of the following mechanism:

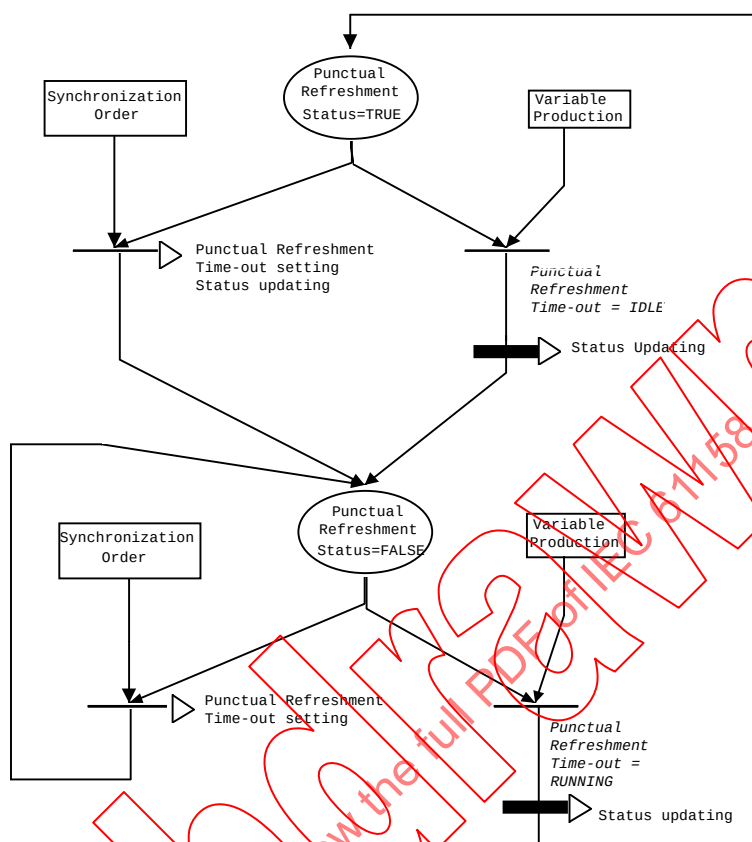


Figure 80 – Punctual refreshment status evaluation net

Initial state: Punctual refreshment status = FALSE

Refreshment time-out state = IDLE

Requests: Variable Production:
A_WRITELOC, A_WRITEFAR or A_WRITE
service invocation by the user

Refreshment time out expiration:
Transition of the time-out associated with the
punctual refreshment evaluation mechanism from
the state *RUNNING* to the state *IDLE*

Synchronization order:
Occurrence of the synchronization order
associated with the punctual refreshment
elaboration mechanism.

Action: Refreshment time-out setting:
Transition of the time-out associated with the
punctual refreshment mechanism from the state
IDLE to the state *RUNNING*, with a preset equal to
the *Production Time Slot* attribute value.

Status updating:
Providing of the asynchronous refreshment status
value to the network.

13.2.1.3.4 Basic services

The variable access basic services in the MPS environment allow the user to access the associated application objects identified by their specification.

The variable specification corresponds:

- either to the application name for variables accessed in a global way. In this case the application name corresponds to the *A_Name* of the *Consumed Variable Local Image* or *Produced Variable Local Image* or *Third part Variable Local Image* objects.
- or to the partial explicit access name either for structured variables or array type variables partially accessible. This access name is built together with the application variable *A_Name* followed by the access path corresponding to the successive enumeration of field names of the tree of the structure and/or index elements of an array.
- or to the access path in the case of partial or global renamed access on a variable. This name corresponds to the *A_Name* attribute of the Variable Access object

13.2.1.3.4.1 Local read service

13.2.1.3.4.1.1 Functionality

With this service the user can read a variable value or a field of a structured variable value or the element of an array type variable value which is available in the local communication entity.

13.2.1.3.4.1.2 Service primitives

A local read is performed by using A_READLOC service primitives:

A_READLOC.Rq (Argument): Local read request

A_READLOC.Cnf (Result): Local read confirmation

Table 312 – A_Readloc service parameters

PARAMETER NAME	Rq	Cnf
Argument		
Variable specification	M	M(=)
Result (+)		S
Value		M
Refreshment status		C
Refreshment status		C
Promptness status		C
Punctual Promptness status		C
Result (-)		S
Type of error		M

Argument:

Information associated with a local read request.

Variable specification:

Corresponds to the identification of a *Produced Variable Local Image* object in the MPS addressing space.

Result (+):

Valid answer associated with a local read request

Value:

It is the value of the variable, or field, or array element specified in the request. This parameter corresponds to the attribute *Public value* of the *Consumed Variable Local Image* object for the non resynchronized variables. For resynchronized variables this parameter corresponds to the attribute *Private value* of the *Consumed Variable Local Image* object.

Refreshment status:**Punctual Refreshment status:**

Information related to the production validity.

This information come from the attribute *Transmitted status value* of the object *Consumed Variable Local Image*.

Promptness status:

Punctual Promptness status:

Information related to the consuming validity.

This information came from the attributes with same name within the *Consumed Variable Local Image* object.

Result (-):

This indicates that the local read service failed.

Type of error:

Various error types may be returned in the confirmation of a read request .

Error_1: The variable value is temporarily not accessible.

Error_2: The variable value is not accessible by means of this service. It corresponds to a variable which is not locally declared, or which is accessible only by production services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

13.2.1.3.4.1.3 Service procedure

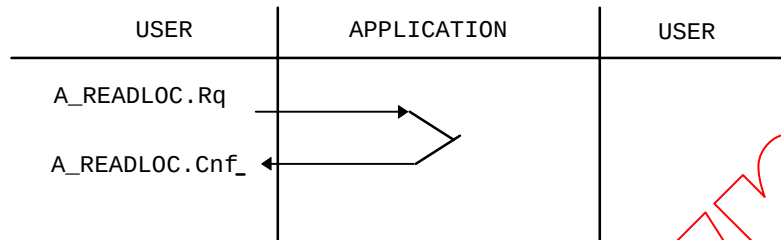


Figure 81 – A_Readloc service procedure

This service ensures the value integrity of the variable or the field or the element handled.

The value integrity of a variable is ensured in a read operation when all the binary elements, that is to say all the binary elements of the data associated with this value come from the same update.

Furthermore requests on a variable are processed sequentially: a new request may be made only after reception of confirmation upon the previous one.

13.2.1.3.4.2 Local write service

13.2.1.3.4.2.1 Functionality

With this service the user can write a variable value or a field of a structured variable value or an element of an array type variable value in the local communication entity.

13.2.1.3.4.2.2 Service primitives

A local write is performed using A_WRITELOC service primitive.

A_WRITELOC.Rq (Argument) : Local write request.

A_WRITELOC.Cnf (Result) : Local write confirmation.

Table 313 – A_Writeloc service parameters

PARAMETER NAME	Rq	Cnf
Argument		
Variable specification	M	M(=)
Value	M	
Result (+)		S
Result (-)		S
Type of error		M

Argument:

Information associated with a local write request

Variable Specification:

Corresponds to the identification of a *Consumed Variable Local Image* object in the MPS addressing space. The definition of this parameter corresponds to the one shown in paragraph.3.8

Value:

Value to be written to the specified variable specification, or field of a structured variable, or element of an array type variable.

This parameter corresponds to the attribute *Public* value of the *Produced Variable Local Image* object for the non resynchronized variables. For the resynchronized variables this parameter corresponds to the attribute *Private* value of the *Produced Variable Local Image* object.

Result (+):

No parameters

Result (-):

This indicates that the local write service failed.

Type of error:

Various error types may be returned in the confirmation of a read request .

Different error types can be returned with the local write confirmation .

Error_1: The variable value is temporarily not accessible.

Error_2: The variable is not accessible with use of this service. It corresponds to a variable which is not locally declared, or which is accessible only by consuming services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

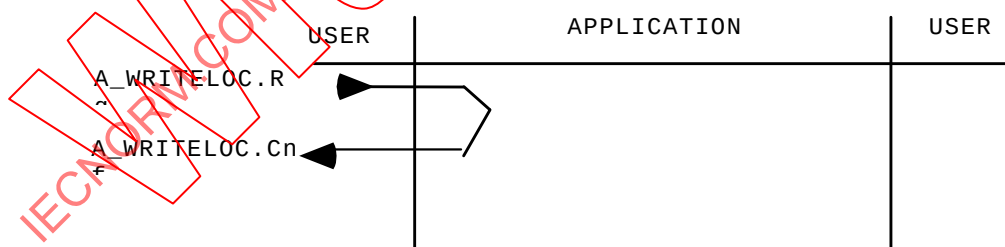
13.2.1.3.4.2.3 Service procedure

Figure 82 – A_Writeloc service procedure

This service ensures the integrity of the value of the handled variable.

The integrity of the written variable corresponds to provide the network with a consistent value coming from the same write operation.

All status elaborated by the producing entity and provided to the network must be consistent with the value of the transmitted variable.

Furthermore this service triggers the mechanism associated with the service elements which elaborate refreshment and punctual refreshment status of the *Produced Variable Local Image* object.

Partial access on a variable when writing a value triggers mechanism of production validity status elaboration associated with this variable.

Furthermore request on variable are processed sequentially; a new request can be made only after reception of confirmation upon the previous one.

13.2.1.3.4.3 Updating service

13.2.1.3.4.3.1 Functionality

This service supports the request (to the bus arbitrator) for the updating of a variable value at the consuming entities level. The variable value is locally available in the producer communication entity level.

This service can be used by the application entity which produces the variable, by an application entity which consumes it, or by an application entity which is neither producer nor consumer (ie: a third party entity).

13.2.1.3.4.3.2 Service Primitives

An updating request is performed with the following service primitives:

A_UPDATE.Rq (Argument) : Updating request.

A_UPDATE.Cnf (Result) : Updating confirmation.

Table 314 – A_Update service parameters

PARAMETER NAME	Rq	Cnf
Argument		
Variable specification	M	M(=)
Priority	M	
Result (+)		S
Result (-)		S
Type of error		M

Arguments:

Information associated with an updating request.

Variable Specification:

Corresponds to the identification of a *Produced Variable Local Image* object or a *Produced Variable Local Image* object or *Third party Variable Local Image* object in the MPS addressing space

Priority:

This parameter allows the user to choose between two flow control possibilities **Urgent** and **Normal** to request the updating of a variable the name of which has been specified in the request.

The flow control ensures that an urgent request, will trigger an updating before a subsequent normal request .

Result (+):

No parameters.

Result (-):

This indicates that the updating service failed.

Type of error:

Various error types may be returned in the service confirmation:

Error_1: The variable is not declared at the AP level of the request initiator.

Error_2: The request cannot be accepted given the local limits concerning the number of outstanding requests.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

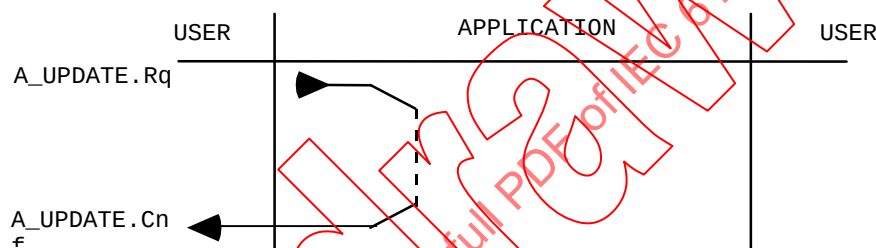
13.2.1.3.4.3.3 Service procedure

Figure 83 – A Update service procedure

The service confirmation is a positive one when the request is positively confirmed at the data link level.

Requests may be chained together even before receiving all confirmations. However; confirmations are not always issued in the order of requests previously initiated.

This service guarantees that the user will receive a confirmation after a request.

For a SYNCHRONIZATION class variable, this service can be initiated only by the variable producer.

NOTE The maximum number of outstanding requests depends on the implementation, and saturation is specified in error codes associated with the confirmation.

13.2.1.3.4.4 Remote read service**13.2.1.3.4.4.1 Functionality**

With this service the consumer user of a variable can take the initiative of an information exchange on the bus, before performing a read on the new variable value locally available.

13.2.1.3.4.4.2 Service Primitives

Remote read is done with the A_READFAR service primitive:

A_READFAR.Rq (Argument): Remote read request

A_READFAR.Cnf (Result): Remote read confirmation with return of the read value

Table 315 – A_Readfar service parameters

PARAMETER NAME	Rq	Cnf
Argument		
Variable specification	M	M(=)
Priority	M	
Result (+)		S
Value		M
Refreshment Status		C
Punctual Refreshment Status		C
Promptness Status		C
Punctual Promptness status		C
Result (-)		S
Type of error		M

Argument:

Information associated with a remote read request.

Variable specification:

Corresponds to the identification of a *Consumed Variable Local Image* object in the MPS addressing space..

Priority:

This parameter allows the user to choose between two flow control possibilities **Urgent** and **Normal** to request the updating of a variable the name of which has been specified in the request.

The flow control guarantees that an urgent request, will trigger an updating before a subsequent normal request .

Result (+):

This result indicates that the requested service has succeeded.

Value:

The value of the variable or field, or a array element specified in the request.

This parameter corresponds the attribute *Public value* of the object *Consumed Variable Local Image* for the non resynchronized variables. For resynchronized variables this parameter corresponds to the attribute *Private value* of the object *Consumed Variable Local Image*.

Refreshment status:

Punctual Refreshment status:

Information relative to the production validity This information comes from the attribute *Transmitted status value* of the *Consumed Variable Local Image* object.

Promptness status:

Punctual Promptness status:

Information relative to the consumption validity

This information comes from the attributes with the same name within the object *Consumed variable Local Image*.

Result (-):

This indicates that the requested service failed.

Type of error:

Various error types may be returned in the service confirmation:

Error_1: The variable value is temporarily not accessible.

Error_2: The variable value is not accessible by means of this service. It corresponds to a variable which is not locally declared or which is accessible only by production services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

Error_4: The request cannot be accepted given the local limits concerning the number of outstanding requests .

Error_5: The request has been aborted due to expiration of the time-out associated with the reception waiting of the variable value.

13.2.1.3.4.4.3 Service procedure

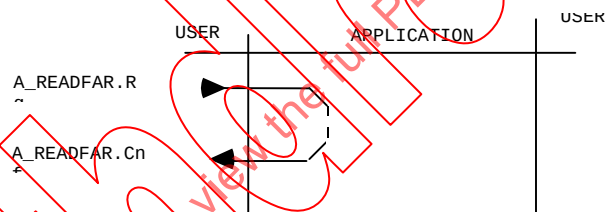


Figure 84 – A_Readfar service procedure

This service guarantees the integrity of the handled variable value, or field, or array element value.

Requests may be chained together. A new request can be issued without waiting for corresponding confirmations. However, confirmations are not always issued in the same order as the requests.

This service guarantees that the user will receive a confirmation after a request.

The value integrity of a variable value is ensured in a read operation when all the binary elements of the data associated with this value come from the same updating.

13.2.1.3.4.5 Remote write service

13.2.1.3.4.5.1 Functionality

With this service ,the variable producer user may initiate an information exchange on the bus after providing the variable value to the network.

13.2.1.3.4.5.2 Service Primitives

Remote write is performed by using the A_WRITEFAR service:

A_WRITEFAR.Rq (Argument): Remote write request

A_WRITEFAR.Cnf (Result): Remote write confirmation

Table 316 – A_Writefar service parameters

PARAMETER NAME	Rq	Cnf
Argument		
Variable specification	M	M(=)
Priority	M	M(=)
Value	M	
Result (+)		S
Result (-)		S
Type of error		M

Arguments:

Information associated with a remote request

Variable specification:

Corresponds to the identification of a *Produced Variable Local Image* object in the MPS space

Priority:

This parameter allows to the user to choose between two flow control possibilities **Urgent** and **Normal** to request the updating of a variable which the name of which has been specified in the request.

The flow control ensures that an urgent request, will trigger an updating before a subsequent normal request .

Value:

Upon a remote write the user associates with the variable specification the value he intends to affect it.

This parameter corresponds to the attribute *Public value* of the *Produced Variable Local Image* object for the non resynchronised variables. For the resynchronized variables this parameter corresponds to the attribute *Private value* of the *Produced Variable Local Image* object.

Result (+):

No parameters.

Result (-):

This indicates that the remote write service failed.

Type of error:

Different error types may be returned in the service confirmation:

Error_1: The variable value is temporarily not accessible.

Error_2: The variable is not accessible by means of this service. It corresponds to a variable which is not locally declared or which is accessible only by consuming services.

Error_3: The variable is declared invalid at the network management level with reference to consistency checking of the data base.

Error_4: The remote write request cannot be accepted due to local limits concerning the number of outstanding requests. However, the local variable value has been updated.

Error_5: The request has been withdrawn

13.2.1.3.4.5.3 Service procedure

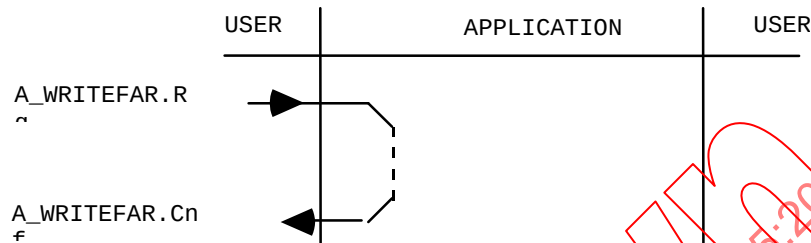


Figure 85 – A_Writefar service procedure

This service ensures the value integrity of the handled variable or the field, or the element .

Requests may be chained together. A new request can be issued without waiting for corresponding confirmations. However, confirmations are not always issued in the same order as the requests.

This service ensures that the user will receive a confirmation after a request.

The integrity of a written variable value corresponds to an updating of the network with a consistent value coming from only one write operation. This service also ensures the consistency between the produced variable value and statuses related to the production validity which are optionally associated.

Moreover, this service triggers the mechanisms associated with the service elements that elaborate the value of the refreshment status and the punctual refreshment status of the *Produced Variable Local Image* object.

NOTE The value provided to the consuming entities will be the one which was available at the producer level during its network updating.

13.2.1.3.4.6 Transmission indication service

This service is optional, and indicates to a producer user that a variable has been transmitted on the network.

13.2.1.3.4.6.1 Service primitives

A transmission indication is performed using the A_SENT service primitive

A_SENT.Ind (Result): Transmission indication of a variable value