TECHNICAL SPECIFICATION

IEC 61158-5

First edition
1999-03

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

Part 5: Application Layer Service definition

Reference number
IEC 61158-5:1999(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.

TECHNICAL SPECIFICATION

IEC 61158-5

First edition
1999-03

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

Part 5: Application Layer Service definition

International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland
Telefax: +41 22 919 0300 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

Clause Page

Clause Page

Clause Page

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 co-operation 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 produced 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 technical specification may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

The main task of IEC technical committees is to prepare International Standards. In exceptional circumstances, a technical committee may propose the publication of a technical specification when

- the required support cannot be obtained for the publication of an International Standard, despite repeated efforts, or
- The subject is still under technical development or where, for any other reason, there is the future but no immediate possibility of an agreement on an International Standard.

Technical specifications are subject to review within three years of publication to decide whether they can be transformed into International Standards.

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

The text of this technical specification is based on the following documents:

Enquiry draft	Report on voting
65C/199/FDIS	65C/207+207A/RVD

Full information on the voting for the approval of this technical specification 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)

Annexes C to L form an integral part of this technical specification.

Annexes A and B are for information only.

This publication will be reviewed by the committee responsible for its preparation before 2002. Information relating to confirmation, amendment or revision of the publication is available from the IEC web site.

A bilingual version of this technical specification may be issued at a later date.

INTRODUCTION

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

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

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

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 technical specification 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 specification 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.

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

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

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 7498–4:1989, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management framework*

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)*

ISO/IEC 8824–1:1995, *Information technology – Abstract Syntax Notation One (ASN.1); Specification of basic notation*
Amendment 1(1996): Rules of extensibility

ISO/IEC 8824–2:1995, *Information technology – Abstract Syntax Notation One (ASN.1); Information object specification*
Amendment 1(1996): Rules of extensibility

ISO/IEC 8824–3:1995, *Information technology – Abstract Syntax Notation One (ASN.1) Constraint specification*

ISO/IEC 8824–4:1995, *Information technology – Abstract Syntax Notation One (ASN.1) Parameterization of ASN.1 specifications*

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*

IEC 61158–3:1999, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems– Part 3: Data Link Layer service definition*

IEC 61158–4:1999, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 4: Data Link Layer Protocol Specification*

IEC 61158–6:1999, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 6: Application Layer Protocol Specification*

ANSIIEEE 754:– *IEEE Standard for 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

3.6 FAL Specific Definitions

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

3.6.1 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

3.6.2 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.

3.6.3 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

3.6.4 Application Relationship Application Service Element Definition

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

3.6.5 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

3.6.6 Client Definition

role of an AR endpoint in which it issues confirmed service request APDUs to a single AR endpoint acting as a server

3.6.7 Conveyance Path Definition

unidirectional flow of APDUs across an application relationship

3.6.8 Dedicated AR Definition

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

3.6.9 Dynamic AR Definition

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

3.6.10 Error Class Definition

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

3.6.11 Error Code Definition

identification of a specific type of error within an error class

3.6.12 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

3.6.13 Application Layer Interoperability Definition

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

3.6.14 Management Information Definition

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

3.6.15 Peer Definition

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

3.6.16 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

3.6.17 Pre-established AR Endpoint Definition

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

3.6.18 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

3.6.19 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

3.6.20 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

3.6.21 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

3.6.22 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.

3.6.23 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

3.6.24 Pull Subscriber Definition

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

3.6.25 Push Subscriber Definition

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

3.6.26 Resource Definition

resource is a processing or information capability of a subsystem

3.6.27 Server Definition

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

3.6.28 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

3.7 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
AR	Application Relationship
AREP	Application Relationship End Point
ASE	Application Service Element
CIM	Computer Integrated Manufacturing
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLCEP	Data Link Connection End Point
DLSAP	Data Link Service Access Point
FAL	Fieldbus Application Layer
FIFO	First In First Out
ID	Identifier
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
LME	Layer Management Entity
OSI	Open Systems Interconnect
PDU	Protocol Data Unit
PL	Physical Layer
SAP	Service Access Point
SDU	Service Data Unit
SMIB	System Management Information Base

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 technical specification. 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 1 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.

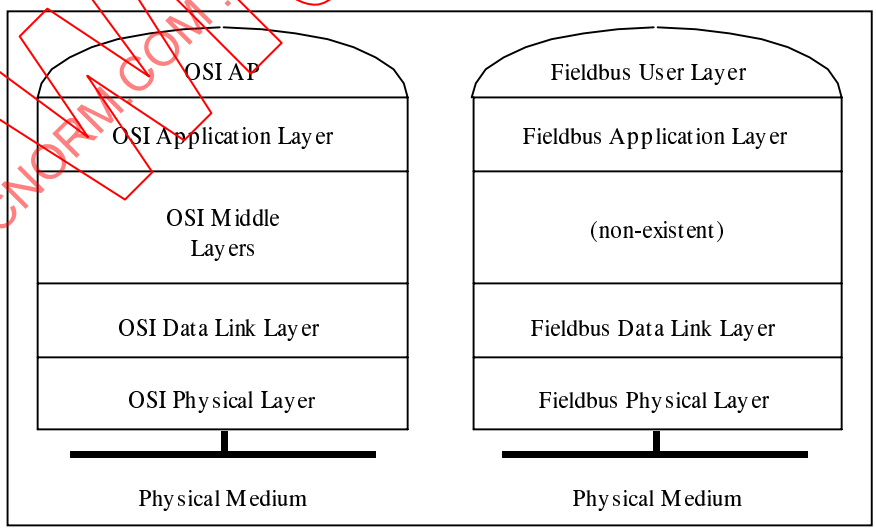


Figure 1 – Relationship to the ISO OSI Reference Model

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.

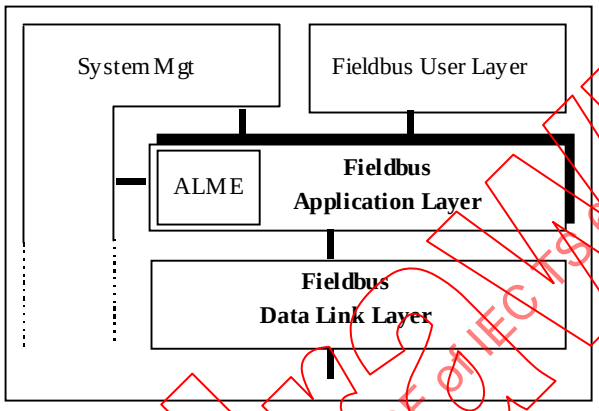


Figure 2 – Architectural Positioning of the Fieldbus Application Layer

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 subclause 4.2 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 technical specification. 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 specification 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 the figure below. 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.

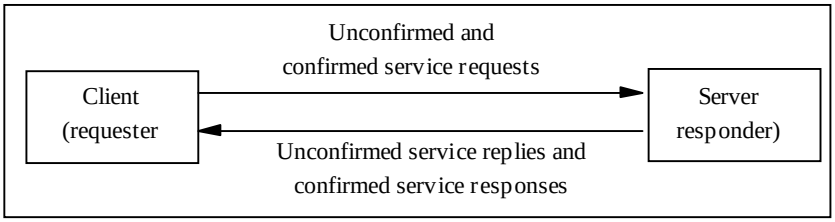


Figure 3 – Client/Server Interactions

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 specification.

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.

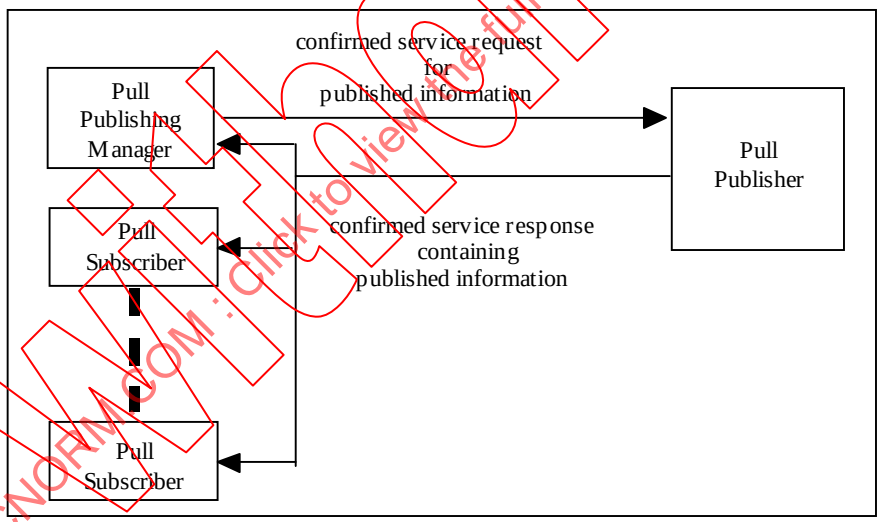


Figure 4 – Pull Model Interactions

4.2.3.3.2.2 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.

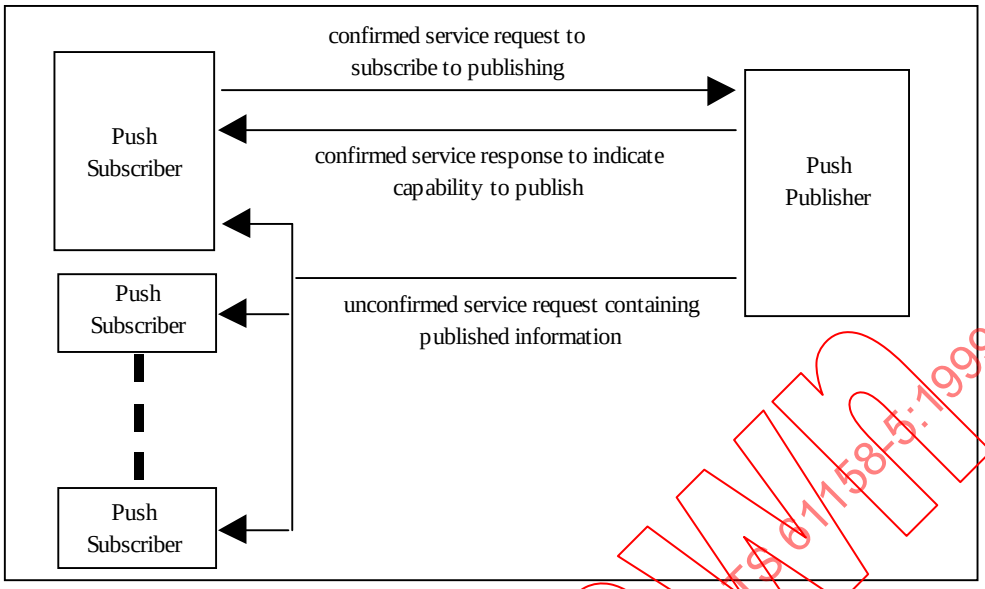


Figure 5 – Push Model Interactions

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.

Type	Description
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 technical specification. 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 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 the figure below, are provided by the FAL communications entity of the AP, known as the FAL Applications Entity (FAL AE).

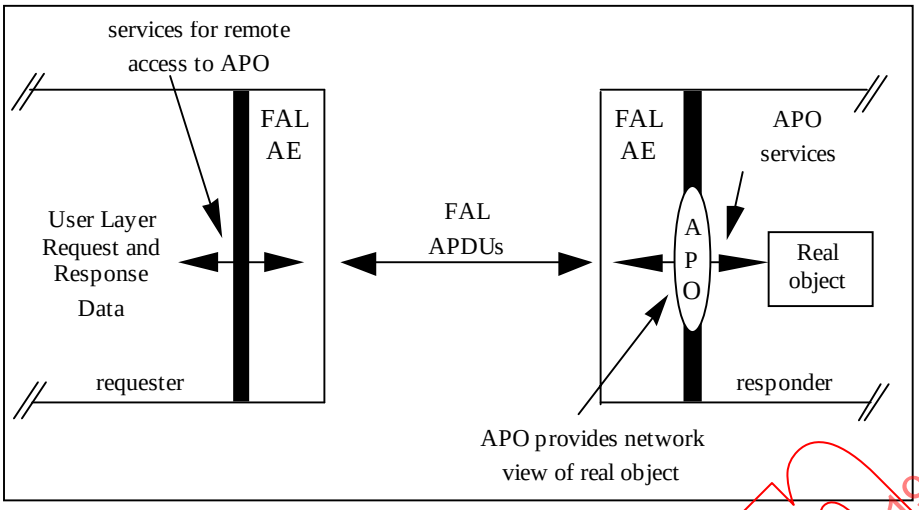


Figure 6 – APOs Services Conveyed by the FAL

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 technical specification 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 technical specification.

4.2.4.3 APOs as Instances of APO Classes

APO Classes are defined in this technical specification 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 technical specification 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.

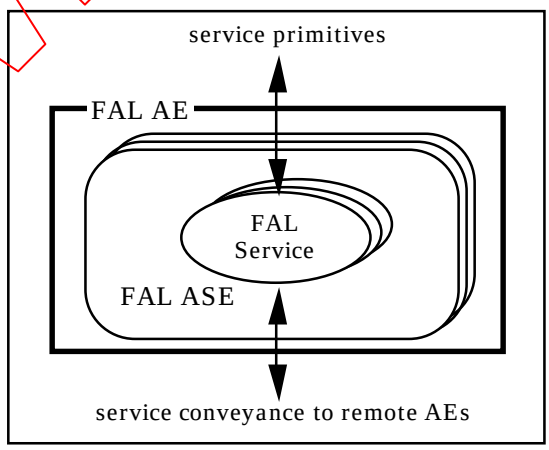


Figure 7 – Application entity structure

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 Introduction

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 technical specification.

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 the FAL ASEs and their architectural relationships.

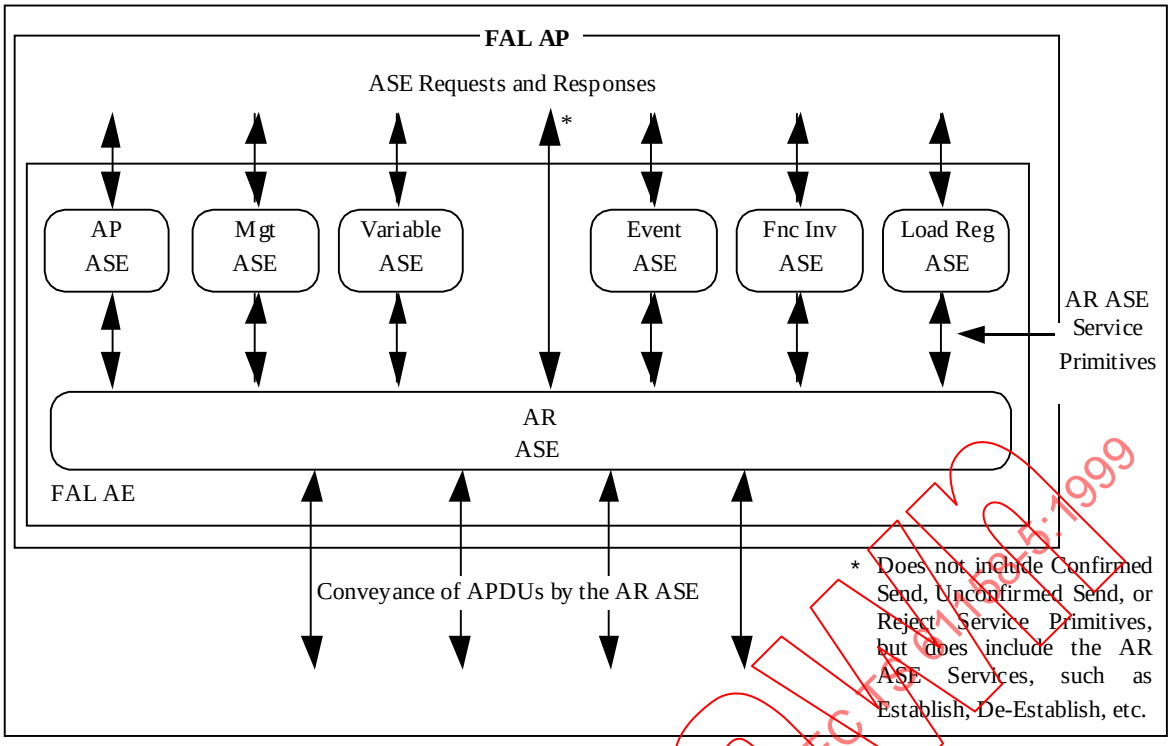


Figure 8 – FAL ASEs

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 though the FAL. The specific operational services that apply to each object type are specified in the definition of the ASE for the object type. The figure below illustrates the integration of management and operational services for an object within an AP.

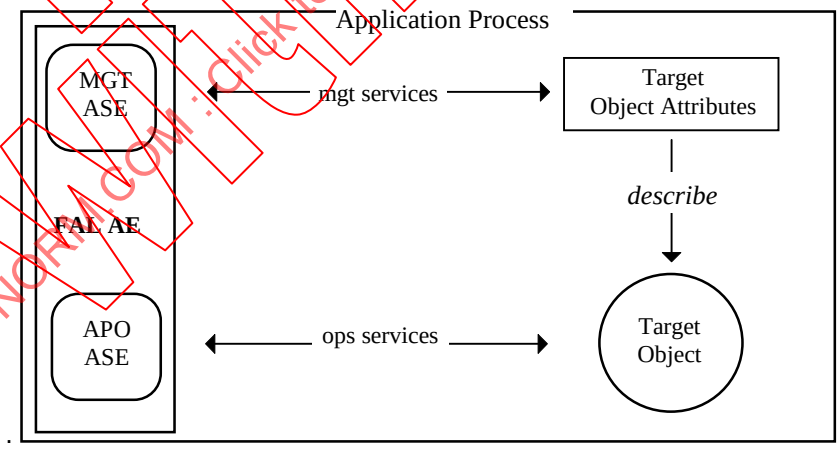


Figure 9 – FAL Management of objects

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.

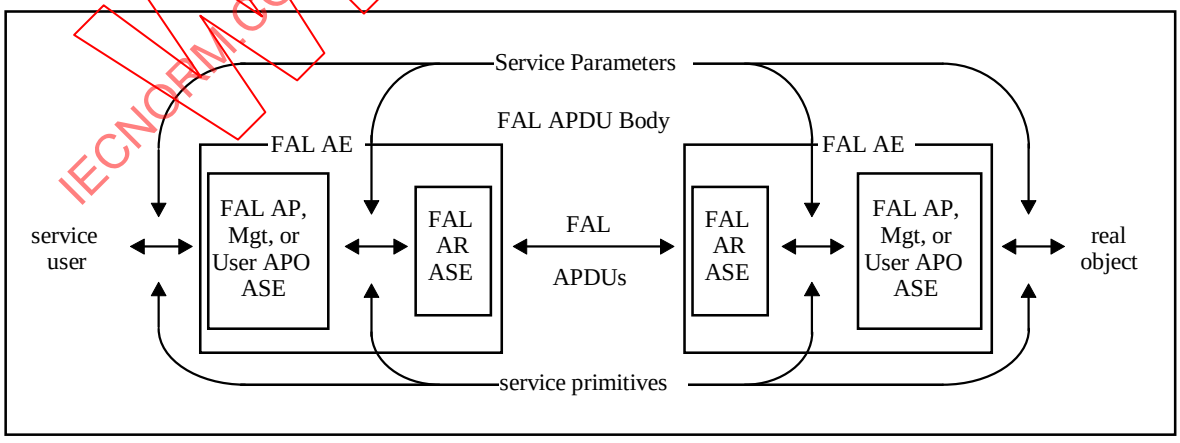


Figure 10 – ASE Service Conveyance

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 must be provided for the FAL.

The mechanism used by the FAL has two parts. First, all FAL ASEs share a common FAL APDU header. This header identifies the abstract syntax for the body of each FAL APDU. Second, the transfer syntax rules are fixed, not negotiated, for all exchanges over a particular AR. Therefore, the presentation context responsibility for identifying the set of transfer syntax rules that were used for each received APDU is not necessary

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 specification. 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 2.2.3.3.2.3.

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).

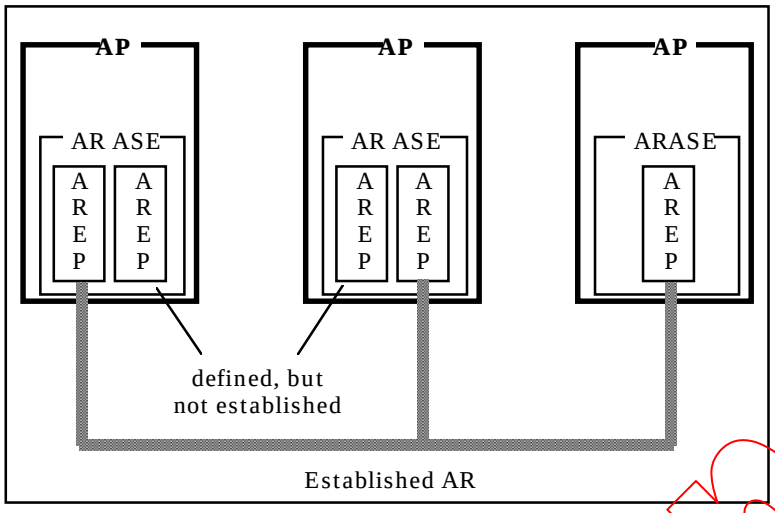


Figure 11 – Defined and Established AREPs

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 though 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 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.

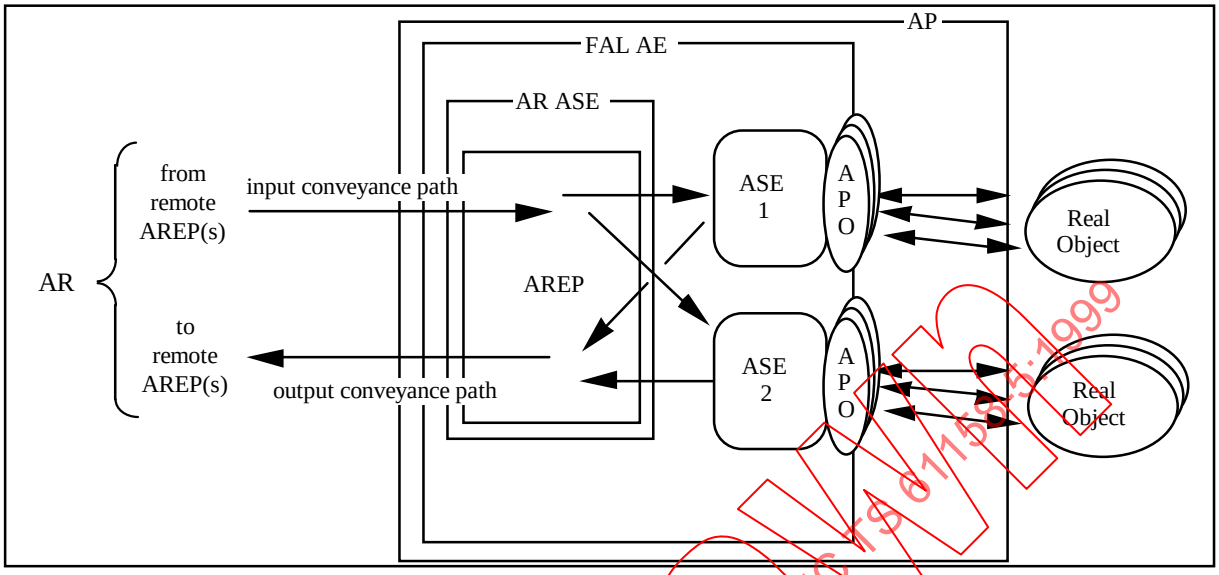


Figure 12 – FAL Architectural Components

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

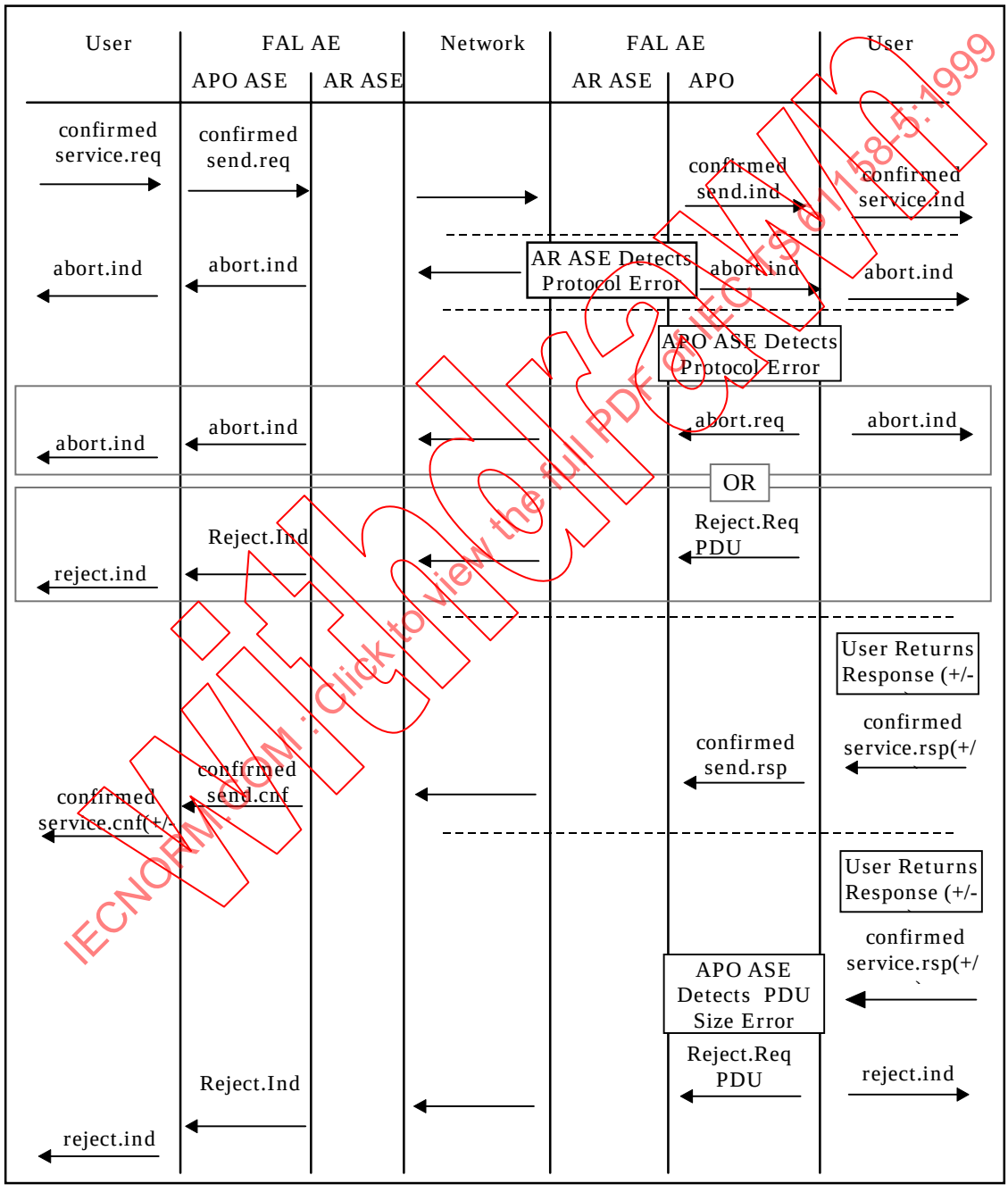


Figure 13 – Time Sequence Diagram For The Confirmed Services

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

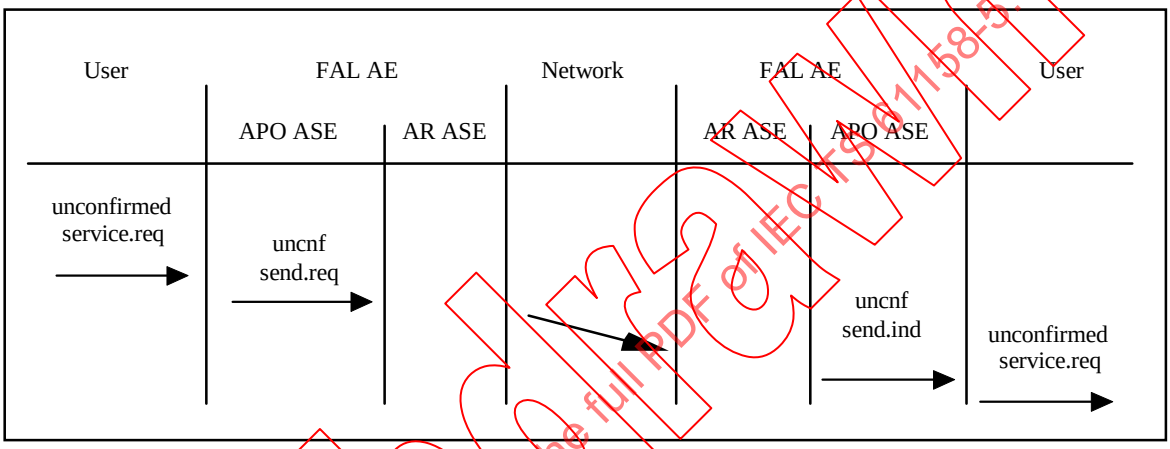


Figure 14 – Time Sequence Diagram For Unconfirmed Services

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	(m)	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 attribute contains user defined descriptive information about the object.

Object Revision

This 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.

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 unconsistent			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.

5 Object Management ASE

5.1 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.

5.2 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 technical specification. Each FAL APO Class defined is described by a set of attributes and a set of services.

5.3 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.

5.3.1 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.

5.3.1.1 Service Parameters

The service parameters for this service are shown in table 2.

Table 2 – Create Service Parameters

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Invoke ID	M	M(=)		
FAL ASE / 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.

5.3.1.2 Service Procedure

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

5.3.2 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 obect is actually deleted or simply removed from network accessibility is a local matter.

5.3.2.1 Service Parameters

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

Table 3 – 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.

5.3.2.2 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.

5.3.3 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.

5.3.3.1 Service Parameters

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

Table 4 – 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. 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 the attributes that are being requested. 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.

5.3.3.2 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.

5.3.4 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.

5.3.4.1 Service Parameters

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

Table 5 – 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.

5.3.4.2 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.

5.3.5 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.

5.3.5.1 Service Parameters

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

Table 6 – 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.

5.3.5.2 Service Procedure

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

5.3.6 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.

5.3.6.1 Service Parameters

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

Table 7 – 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.

5.3.6.2 Service Procedure

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

6 Application Process ASE

6.1 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.

6.2 AP Class Specification

6.2.1 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:	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 Objects (Data Types)
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 Lists 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
4	(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

6.2.1.1 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

6.2.1.2 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 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.

6.2.1.3 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.

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 AR.

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 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.

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.

6.2.1.4 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.

6.3 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

6.3.1 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 technical specification.

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.

6.3.1.1 Service Primitives

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

Table 8 – 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 Jitter	U	U(=)		
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.

6.3.1.2 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.

6.3.2 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.

6.3.2.1 Service Primitives

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

Table 9 – 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.

6.3.2.2 Service Procedure

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

6.3.3 Get Status Service

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

6.3.3.1 Service Primitives

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

Table 10 – 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.

6.3.3.2 Service Procedure

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

6.3.4 Status Notification Service

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

6.3.4.1 Service Primitives

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

Table 11 – 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.

6.3.4.2 Service Procedure

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

6.3.5 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.

6.3.5.1 Service Primitives

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

Table 12 – 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 Definition 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.

6.3.5.2 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.

6.3.6 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.

6.3.6.1 Service Primitives

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

Table 13 – 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 ≥ Max Outstanding Services Receiving
Connection State Conflict	INITIATE-REQ_PDU received from the AR ASE
Service Error	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.

6.3.6.2 Service Procedure

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

6.3.7 Conclude Service

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

6.3.7.1 Service Primitives

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

Table 14 – 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.

6.3.7.2 Service Procedure

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

6.3.8 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.

6.3.8.1 Service Primitives

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

Table 15 – 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.

6.3.8.2 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 Application Relationship ASE

7.1 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 the AR Endpoint object defined in 7.2.

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. The figure below illustrates this concept.

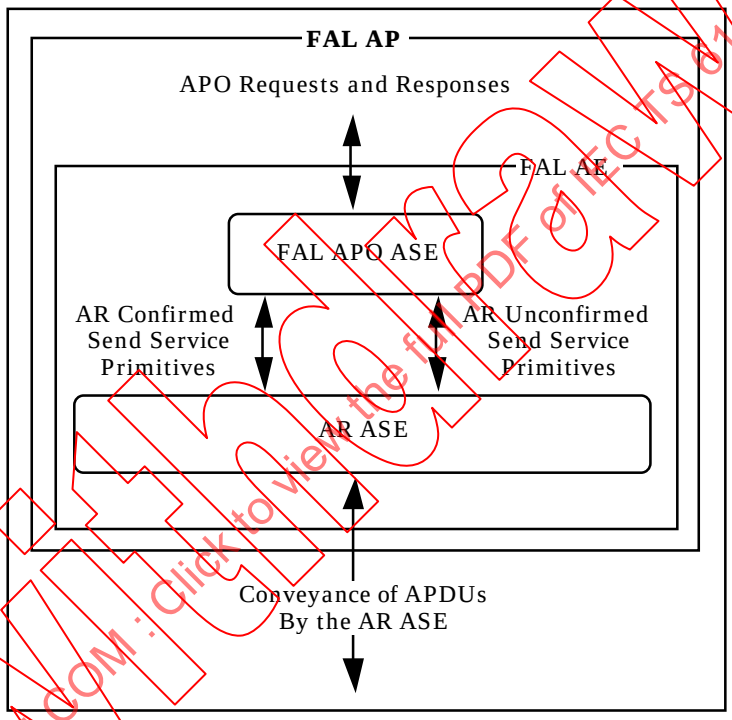


Figure 15 – The AR ASE Conveys APDUs between APs

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.

7.1.1 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.

7.1.1.1 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.

Tables 16 and 17 summarize the characteristics and combinations of each of the AREP roles.

Table 16 – 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 17 – 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								
Client		Yes	Yes					
Server			Yes					
Peer			Yes					
Publisher / Subscriber								
Push Publisher					Yes			
Push Subscriber								
Pull Publ Mgr							Yes	
Pull Publisher								Yes
Pull Subscriber								

7.1.1.2 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.

7.1.1.3 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.

7.1.1.4 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.

7.1.1.4.1 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.

7.1.1.4.2 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.

7.1.1.4.3 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.

7.1.2 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.

7.1.2.1 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.

7.1.2.2 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.

7.1.2.3 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.

7.1.2.4 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.

7.1.2.5 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.

7.1.2.6 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).

7.1.2.7 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 subcriber AR endpoints support 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.

7.1.2.8 Duplicate FAL PDU detection

AR endpoints may be defined to support detection of the receipt of duplicate FAL APDUs by the Data Link Layer. An indication is received from the Data Link Layer 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.

7.1.3 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”.

7.1.4 Application Relationship Classes

AREPs are defined with a combination of characteristics to form different classes of ARs. The descriptions for each class are provided in annexes to this specification. Additional annexes (as addenda) may be created to define new classes as necessary.

7.2 Application Relationship Endpoint Class Specifications

7.2.1 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 annex of this specification.

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-Abort
2	(o)	OpsService:	AR-Get DL-Time
3.	(o)	OpsService:	AR-Status

7.2.1.1 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 technical specification 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.

7.2.1.2 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-Abort

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

AR-Get DL-time

This optional service provides access to the Data Link Layer DL-Time service.

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.

7.3 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.

7.3.1 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.

7.3.1.1 Service Primitives

The service parameters for this service are shown below.

Table 18 – 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.

7.3.1.2 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.

7.3.2 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.

7.3.2.1 Service Primitives

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

Table 19 – 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.

7.3.2.2 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.

7.3.3 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.

7.3.3.1 Service Primitives

The service parameters for this service are shown below.

Table 20 – 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.

7.3.3.2 Service Procedure

7.3.3.2.1 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 the figure below:

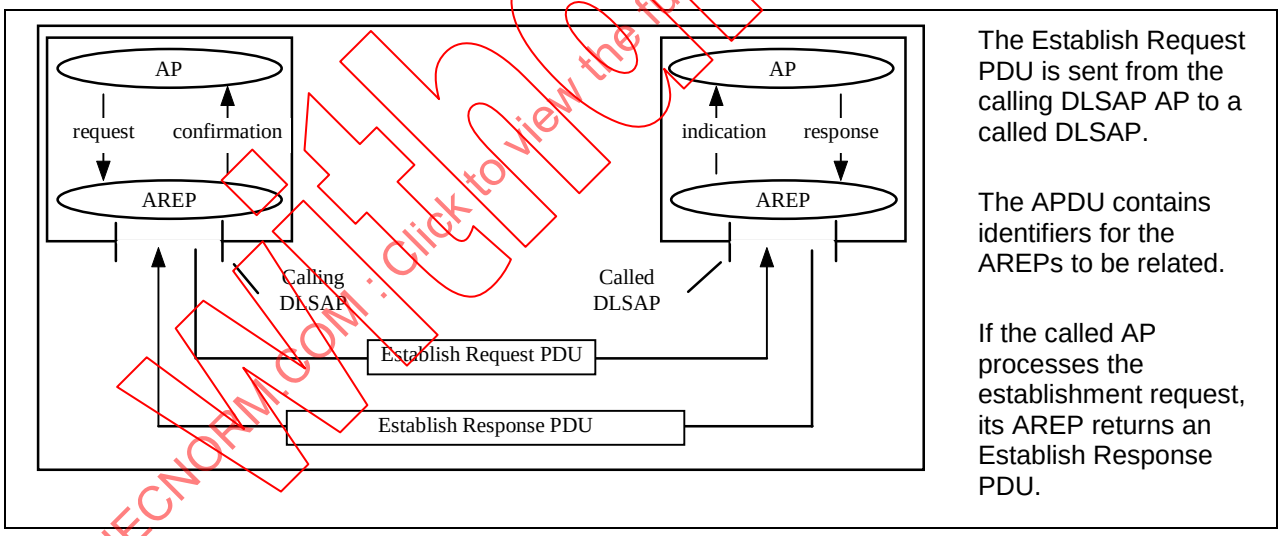
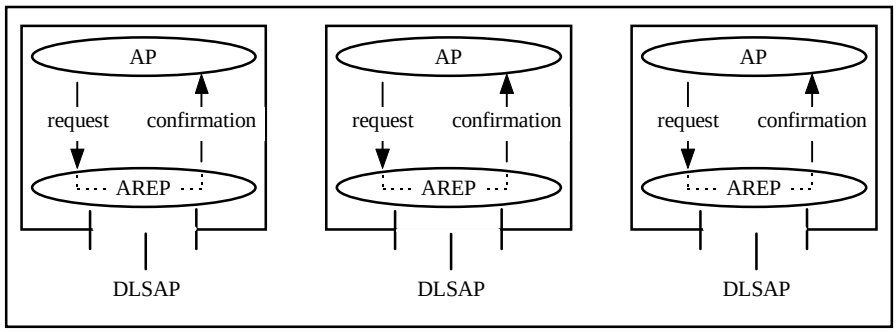


Figure 16 – 1-to-1 AR Establishment

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.

7.3.3.2.2 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 17:



Each AP in the AR establishes its own AREP locally by issuing an Establish request. Establish PDUs are not exchanged.

However, if the AR is supported by a data link connection, DL-PDUs are exchanged

Figure 17 – 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.

7.3.3.2.3 Compatible AREP Classes

The table below 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 21 – 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.

7.3.3.2.4 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.

7.3.4 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.

7.3.4.1 Service Primitives

The service parameters for this service are shown below. The use of this service is for future study.

Table 22 - 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.

7.3.4.2 Service Procedure

The AR-DeEstablish Service is a service that operates through a queue or buffer. Its service procedure is for future study.

7.3.5 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.

7.3.5.1 Service Primitives

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

Table 23 – 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.

7.3.5.2 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.

7.3.6 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.

7.3.6.1 Service Primitives

The service parameters for this service are shown below.

Table 24 – 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.

7.3.6.2 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.

7.3.7 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.

7.3.7.1 Service Primitives

The service parameters for this service are shown below:

Table 25 – 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.

7.3.7.2 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.

7.3.8 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.

7.3.8.1 Service Primitives

The service parameters for this service are shown below.

Table 26 – 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.

7.3.8.2 Service Procedure

This service requests the communication stack to create a scheduled sequence.

7.3.9 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.

7.3.9.1 Service Primitives

The service parameters for this service are shown below.

Table 27 – 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.

7.3.9.2 Service Procedure

This service requests the communication stack to cancel an existing scheduled sequence.

7.3.10 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 compatiblity with the DL-Time service, the service and parameters definitions are not defined in this specification. See IEC 61158-3 for the definition of the DL-Time service.

7.3.11 AR-Status Service

This local service provides the AL user notification of a status change of the AREP.

7.3.11.1 Service Primitives

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

Table 28 – 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.

7.3.11.2 Service Procedure

This service indicates that a significant event, as defined by the status code parameter, occurred in the communication stack.

7.3.12 AR-XON-OFF Service

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

7.3.12.1 Service Primitives

The service parameters for this service are shown below.

Table 29 – 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“.

7.3.12.2 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.

7.3.13 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

7.3.13.1 Service Primitives

The service parameters for this service are shown below.

Table 29 – 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.

7.3.13.2 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.

7.3.14 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.

7.3.14.1 Service Primitives

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

Table 30 – 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.

5.3.14.2 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.

8 Data Type ASE

8.1 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 technical specification.

Data types are defined as instances of the data type class, as shown in the figure 18. Defining new types is accomplished by providing a numeric id and supplying values for the attributes defined for the data type class.

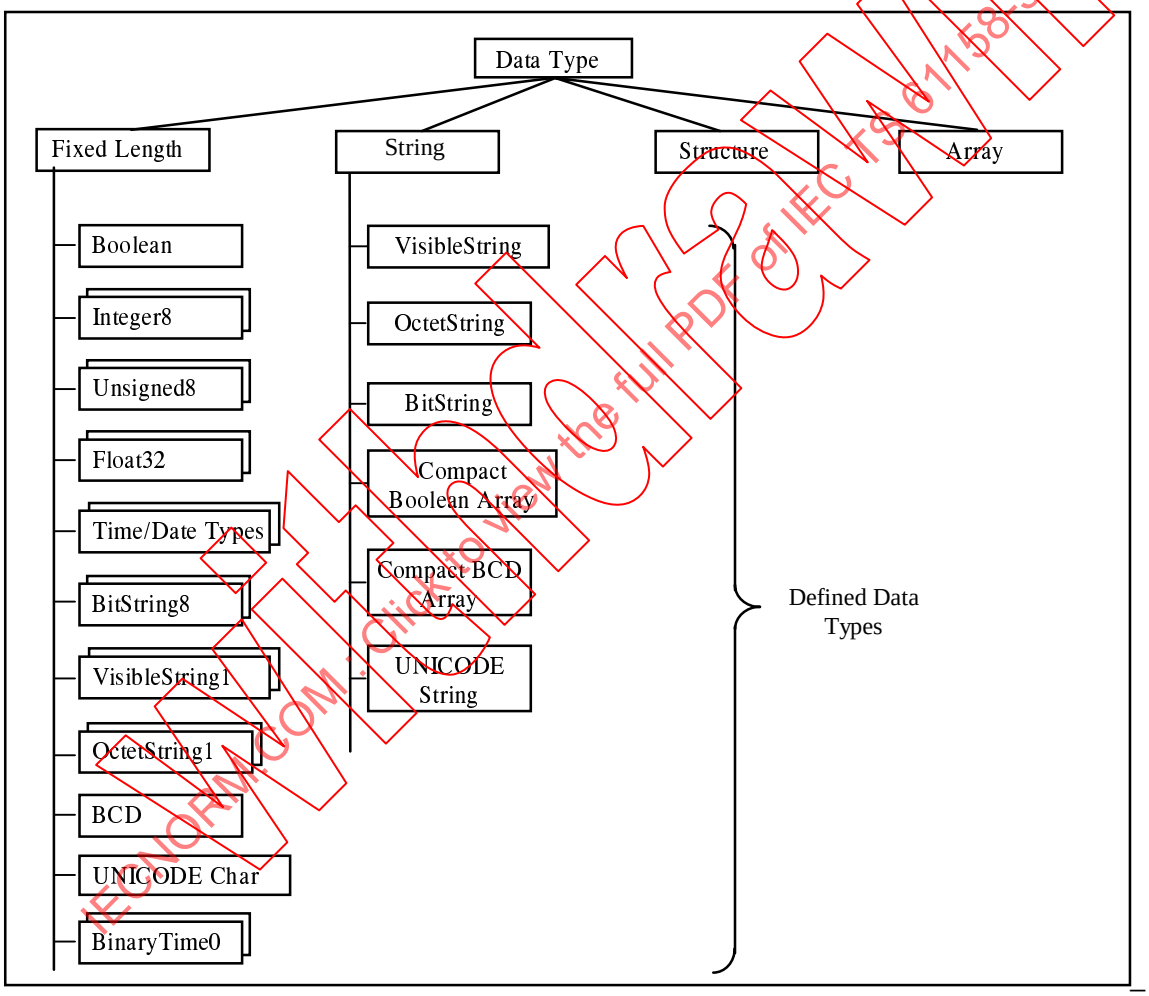


Figure 18 – Data Type Class Hierarchy

The data type definitions in figure 18 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 18.

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.

8.1.1 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 technical specification for the purpose of assigning Fieldbus class identifiers to them. This technical specification does not change their definitions as specified in ISO/IEC 8824.

8.1.1.1 Fixed Length Types

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

8.1.2 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 technical specification, arrays and structures.

8.1.2.1 Strings

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

8.1.2.2 Arrays

An array is composed of an ordered set of homogeneously typed elements. This technical specification 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.

8.1.2.3 Structures

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

8.1.2.4 Nesting level

This technical specification 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".

8.1.3 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 technical specification 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.

8.1.4 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.

8.2 Formal Definition of Data Type Objects

8.2.1 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

8.2.1.1 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

8.3 FAL Defined Data Types

8.3.1 Fixed Length Types

8.3.1.1 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.

8.3.1.2 Integer8

CLASS: **Data Type**

ATTRIBUTES:

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

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

8.3.1.3 Integer16

CLASS: **Data Type**

ATTRIBUTES:

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

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

8.3.1.4 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.

8.3.1.5 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.

8.3.1.6 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.

8.3.1.7 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.

8.3.1.8 Float32

CLASS: **Data Type**

ATTRIBUTES:

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

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

8.3.1.9 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.

8.3.1.10 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.

8.3.1.11 TimeOfDay

CLASS: **Data Type**

ATTRIBUTES:

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

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

8.3.1.12 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.

8.3.1.13 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.

8.3.1.14 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.

8.3.1.15 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.

8.3.2.16 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.

8.3.2.17 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

8.3.2.18 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

8.3.1.19 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.

8.3.1.20 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.

8.3.1.21 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.

8.3.1.22 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.

8.3.1.23 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.

8.3.1.24 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.

8.3.1.25 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.

8.3.1.26 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.

8.3.1.27 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.

8.3.1.28 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.

8.3.1.29 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.

8.3.1.30 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.

8.3.1.31 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.

8.3.1.32 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.

8.3.1.33 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.

8.3.1.34 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.

8.3.1.35 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 µs.

8.3.1.36 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 µs.

8.3.1.37 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.

8.3.1.38 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.

8.3.1.39 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.

8.3.1.40 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.

8.3.2 String Types

8.3.2.1 VisibleString

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	9
2	Data Type Name	=	VisibleString
3	Format	=	STRING
4.1	Octet Length	=	1

This type is defined as the ISO 646 string type.

8.3.2.2 OctetString

CLASS: **Data Type**

ATTRIBUTES:

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

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.

8.3.2.3 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.

8.3.2.4 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.

8.3.2.5 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".

8.3.2.6 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.

8.4 Data type ASE Service Specification

There are no operational services defined for the type object.

9 Variable ASE

9.1 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.

9.1.1 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".

9.2 Variable Model Class Specification

9.2.1 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.

9.2.1.1 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.

9.2.1.2 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 technical specification.

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 technical specification.

9.2.2 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

9.2.2.1 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	Attribute Use
1 Reference to a data type with the format of ARRAY:	Reflects the number of elements defined for the array data type
2 Reference to a data type with the format of FIXED-LENGTH or STRING	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.

9.2.2.2 Services

See the description of the services defined in the Simple Variable class above.

9.2.3 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

9.2.3.1 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.

9.2.3.2 Services

See the description of the services defined for the Simple Variable class above.

9.2.4 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

9.2.4.1 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

9.2.4.2 Services

See the description of the services defined for the Simple Variable class above.

9.3 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.

9.3.1 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.

9.3.1.1 Service Primitives

The service parameters for this service are shown in table 31.

Table 31 – 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.

9.3.1.2 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.

9.3.2 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.

9.3.2.1 Service Primitives

The service parameters for this service are shown in table 32.

Table 32 – 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.

9.3.2.2 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.

9.3.3 Write Service

This confirmed service is used to write the value of a variable. It is not used with the publisher/subscriber model.

9.3.3.1 Service Primitives

The service parameters for this service are shown in table 33.

Table 33 – 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.

9.3.3.2 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.

9.3.4 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.

9.3.4.1 Service Primitives

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

Table 34 – 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.

9.3.4.2 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.

9.3.5 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.

9.3.5.1 Service Primitives

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

Table 35 – 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 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.

9.3.5.2 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.

9.3.6 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.

9.3.6.1 Service Primitives

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

Table 36 – 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.

9.3.6.2 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.

9.3.7 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.

9.3.7.1 Service Primitives

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

Table 37 – 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.

9.3.7.2 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.

9.3.8 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.

9.3.8.1 Service Primitives

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

Table 38 – 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.

9.3.8.2 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.

10 Event ASE

10.1 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.

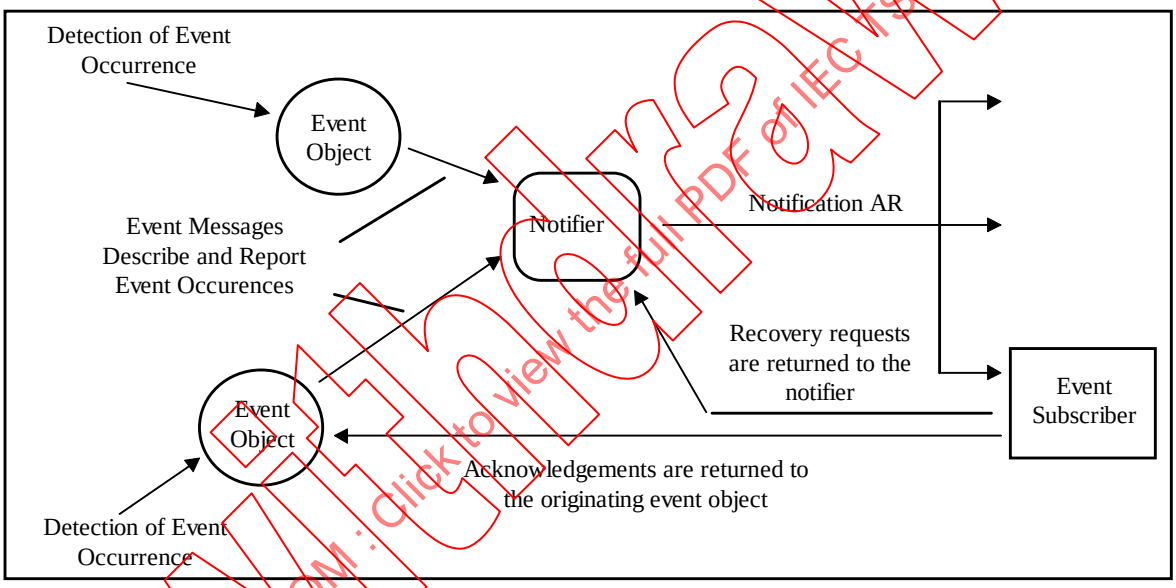


Figure 19 – Event Model Overview

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.

10.2 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.

10.2.1 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

10.2.1.1 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	Event Numeric ID
WITH DATA	Last Detected Event Info.Count
	Event Data Identified by the Event Data Specifier
EVENT DETECTION TIME	Event Numeric ID
WITH DATA	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 rollsover, it rollsover 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 rollsover, or if it rollsover 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.

10.2.1.2 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.

10.2.2 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

10.2.2.1 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.

10.2.2.2 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.

10.2.3 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	

10.2.3.1 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.

10.2.3.2 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.

10.3 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

10.3.1 Acknowledge Event

This service allows the acknowledgment of a reported event.

10.3.1.1 Service Primitives

The service parameters for this service are shown on table 39.

Table 39 –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.

10.3.1.2 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.

10.3.2 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.

10.3.2.1 Service Primitives

The service parameters for this service are shown in table 40.

Table 40 –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.

10.3.2.2 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.

10.3.3 Enable Event Service

This service allows the enabling or disabling of the Event Object.

10.3.3.1 Service Primitives

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

Table 41 – 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.

10.3.3.2 Service Procedure

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

10.3.4 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.

10.3.4.1 Service Primitives

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

Table 42 – 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.

10.3.4.2 Service Procedure

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

10.3.5 Enable Event List Service

This service allows the enabling or disabling of the Event Object.

10.3.5.1 Service Primitives

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

Table 43 – 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.

10.3.5.2 Service Procedure

The Confirmed Service Procedure specified in clause 4 applies to this service. The Enable Eventlist is a "best effort" service.

10.3.6 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.

10.3.6.1 Service Primitives

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

Table 44 – 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.

10.3.6.2 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.

10.3.7 Get Event Summary Service

This confirmed service is used to request summary information for a specified event object.

10.3.7.1 Service Primitives

The service parameters for this service are shown in table 45.

Table 45 – 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.

10.3.7.2 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.

10.3.8 Get Event Summary List Service

10.3.8.1 Service Primitives

The service parameters for this service are shown in table 46.

Table 46 – 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.

10.3.8.2 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.

10.3.9 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 is 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.

10.3.9.1 Service Primitives

The service parameters for this service are shown in table 47.

Table 47 – 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 its 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.

10.3.9.2 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.

11 Load Region ASE

11.1 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.

11.2 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 specification.

11.2.1 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

11.2.1.1 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.

11.2.1.2 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.

11.3 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

11.3.1 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.

11.3.1.1 Service Primitives

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

Table 48 – 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.

11.3.1.2 Service Procedure

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

11.3.2 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.

11.3.2.1 Service Primitives

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

Table 49 – 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.

11.3.2.2 Service Procedure

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

11.3.3 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.

11.3.3.1 Service Primitives

The service parameters for this service are shown in table 50.

Table 50 – 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.

11.3.3.2 Service Procedure

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

11.3.4 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.

11.3.4.1 Service Primitives

The service parameters for this service are shown in table 51.

Table 51 – 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.

11.3.4.2 Service Procedure

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

11.3.5 Discard Service

This confirmed service is used to request that the contents of a load region be discarded.

11.3.5.1 Service Primitives

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

Table 52 – 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.

11.3.5.2 Service Procedure

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

11.4 Load Region State Machines

This document 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 technical specification.

11.4.1 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.

11.4.1.1 Sequencing of Service Primitives

The following table 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 53 - Pull Upload Sequencing of Service Primitives

FAL User(Client)		FAL User(Server) Owner of called Load Region	
Initiate_Load.Ind	←----(T 1)------	Initiate_Load.Req	
Initiate_Load.Req Initiate_Load.Cnf	-------(T2)-----→ ←----(T2)-------	Initiate_Load.Ind Initiate_Load.Rsp	(Transfer started)
Pull_Segment.Req Pull_Segment.Cnf	-------(T3)-----→ ←----(T3)-------	Pull_Segment.Ind Pull_Segment.Rsp	(Segments Transferred)
Terminate_Load.Req Terminate_Load.Cnf	-------(T4)-----→ ←----(T4)-------	Terminate_Load.Ind Terminate_Load.Rsp	(Transfer ended)
Initiate_Load.Rsp	-------(T1)-----→	Initiate_Load.Cnf	

11.4.1.2 Service Parameter Value Constraints

Table 54 summarizes the service parameter constraints.

Table 54 - 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

11.4.1.3 Upload State Transition Table

Table 55 gives the upload state transitions.

Table 55 - 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

11.4.2 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.Req to the Client

11.4.2.1 Sequencing of Service Primitives

The following table summarizes the sequencing of service primitives.

Table 56 - Pull 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 Initiate_Load.Cnf	-------(T2)-----→ ←-----(T2)-------	Initiate_Load.Ind Initiate_Load.Rsp	(Transfer started)
Pull_Segment.Ind Pull_Segment.Rsp	<-------(T3)----- -----(T3)-----→	Pull_Segment.Req Pull_Segment.Cnf	(Transfer continued)
Terminate_Load.Ind Terminate_Load.Rsp	<-------(T4)----- ----(T4)-----→	Terminate_Load.Req Terminate_Load.Cnf	(Transfer ended)
Initiate_Load.Rsp	-------(T1)-----→	Initiate_Load.Cnf	

11.4.2.2 Service Parameter Constraints

Table 57 summarizes the service parameter constraints.

Table 57 - 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

11.4.2.3 Download State Transition Table

Table 58 gives the download state transitions.

Table 58 - Pull Download State Table

Transition	Current State	Event/Action	Next State
1	DOWNLOADABLE	Initiate_Load.Ind &&Upload State = UPLOADABLE Initiate_Load.Rsp(+) Pull_Segment.Req	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.Req	DOWNLOADING
4	DOWNLOADING	Pull_Segment.Cnf(+) &&More Follows=FALSE Store received data Terminate_Load.Req Terminate Reason : = success	DOWNLOAD SUCCESS
5	DOWNLOADING	Pull_Segment.Cnf(-) Discard received data Terminate_Load.Req 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 58 *(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 II 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 II 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

11.4.3 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

11.4.3.1 Sequencing of Service Primitives

Table 59 summarizes the sequencing of service primitives.

Table 59 - 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 Initiate_Load.Cnf	-------(T2)-----→ ←----(T2)-------	Initiate_Load.Ind Initiate_Load.Rsp	(Transfer started)
Push_Segment.Req Push_Segment.Cnf	-------(T3)-----→ ←----(T3)-------	Push_Segment.Ind Push_Segment.Rsp	(Transfer continued)
Terminate_Load.Req Terminate_Load.Cnf	-------(T4)-----→ ←----(T4)-------	Terminate_Load.Ind Terminate_Load.Rsp	(Transfer ended)
Initiate_Load.Rsp	-------(T1)-----→	Initiate_Load.Cnf	

11.4.3.2 Service Parameter Constraints

The following table summarizes the service parameter constraints.

Table 60 - 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

11.4.3.3 Download State Transition Table

Table 61 gives the upload state transitions.

Table 61 - 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 page 197)

Table 61 *(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 II 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

12 Function Invocation ASE

12.1 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.

12.2 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.

12.2.1 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.

12.2.1.1 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

12.2.1.1.1 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 attibute 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.

12.2.1.1.2 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.

12.2.2 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.

12.2.2.1 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

12.2.2.1.1 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.

12.2.2.1.2 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.

12.3 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

12.3.1 Start Service

This confirmed service is used to request that a function invocation be started.

12.3.1.1 Service Primitives

The service parameters for this service are shown in table 62.

Table 62 – 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.