

Distribution automation using distribution line carrier systems

Part 4. Data communication protocols

Section 41: Application protocols Distribution line message specification

The European Standard EN 61334-4-41 : 1996 has the status of a
British Standard

ICS 35.100.70

National foreword

This British Standard is the English language version of EN 61334-4-41 : 1996. It is identical with IEC 61334-4-41 : 1996.

The UK participation in its preparation was entrusted to Technical Committee PEL/57, Power system control and associated communications, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

From 1 January 1997, all IEC publications have the number 60000 added to the old number. For instance, IEC 27-1 has been renumbered as IEC 60027-1. For a period of time during the change over from one numbering system to the other, publications may contain identifiers from both systems.

Cross-references

Attention is drawn to the fact that CEN and CENELEC Standards normally include an annex which lists normative references to international publications with their corresponding European publications. The British Standards which implement these international or European publications may be found in the BSI Standards Catalogue under the section entitled ‘International Standards Correspondence Index’, or by using the ‘Find’ facility of the BSI Standards Electronic Catalogue.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the EN title page, pages 2 to 125 and a back cover.

This British Standard, having been prepared under the direction of the Electrotechnical Sector Board, was published under the authority of the Standards Board and comes into effect on 15 April 1998

© BSI 1998

ISBN 0 580 28702 5

Amendments issued since publication

Amd. No.	Date	Text affected

English version

Distribution automation using distribution line carrier systems
Part 4: Data communication protocols
Section 41: Application protocols
Distribution line message specification
(IEC 1334-4-41:1996)

Automatisation de la distribution à
l'aide de systèmes de communication
à courants porteurs
Partie 4: Protocoles de communication
de données
Section 41: Protocoles d'application
Spécification des messages de ligne
de distribution
(CEI 1334-4-41:1996)

Verteilungsautomatisierung mit Hilfe
von Trägersystemen auf
Verteilungsleitungen
Teil 4: Datenkommunikationsprotokolle
Hauptabschnitt 41:
Anwendungsprotokolle
Verteilungsleitungs-
Nachrichtenspezifikation
(IEC 1334-4-41:1996)

This European Standard was approved by CENELEC on 1996-10-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Foreword

The text of document 57/261/FDIS, future edition 1 of IEC 1334-4-41, prepared by IEC TC 57, Power system control and associated communications, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61334-4-41 on 1996-10-01.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 1997-07-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 1997-07-01

Annexes designated "normative" are part of the body of the standard.

Annexes designated "informative" are given for information only.

In this standard, annexes A and ZA are normative and annex B is informative.

Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 1334-4-41:1996 was approved by CENELEC as a European Standard without any modification.

CONTENTS

	Pag
INTRODUCTION	7
Clause	
1 Scope.....	8
2 Normative references	9
3 Definitions.....	10
3.1 Reference model definitions.....	10
3.2 Service convention definitions	10
3.3 Other definitions.....	11
3.4 Abbreviations	12
3.5 Conventions	12
3.5.1 Service parameter description	12
3.5.2 Addressing in DLMS	13
3.5.3 Service conventions	14
3.5.4 Sending and receiving DLMS user and DLPM	14
3.5.5 Requesting and responding DLMS user	15
3.5.6 Client and server of a service	15
3.5.7 Object modelling	15
4 Virtual distribution equipment	17
4.1 Relationship of the VDE to the OSI model.....	17
4.1.1 The VDE within the AP	17
4.1.2 The AE within the VDE	19
4.2 Relationship with a real distribution device	20
4.3 Structure of a VDE.....	21
4.3.1 Structure overview	21
4.3.2 The VDE-handler.....	22
4.3.3 The data set	23
4.3.4 The task invocation	23
4.3.5 The variable	23
4.4 Specification of DLMS objects	24
4.4.1 Object name	24
4.4.2 Scope of name	24
4.4.3 Scope of access	24
4.4.4 Lifetime	25
4.4.5 Object classes	25
4.4.6 Object Name parameter.....	26
4.4.7 DLMS object description	27
4.5 Conformance description	28
4.5.1 Purpose.....	28
4.5.2 Structure.....	28
4.5.3 Parameters	28
5 Context management services	30
5.1 Introduction	30
5.2 Initiate service	30
5.2.1 Purpose.....	30
5.2.2 Structure	31
5.2.3 Parameters	31
5.2.4 Service procedure.....	33
5.3 Abort service	34
5.3.1 Purpose.....	34
5.3.2 Structure	34
5.3.3 Parameter	34
5.3.4 Service procedure.....	34

6 VDE support services	
6.1 Virtual distribution equipment description	35
6.1.1 The VDE object	35
6.1.2 VDE Content	36
6.1.3 Operations on the VDE	37
6.2 GetStatus service	37
6.2.1 Purpose	37
6.2.2 Structure	37
6.2.3 Parameters	37
6.2.4 Service procedure	38
6.3 GetNameList service	39
6.3.1 Purpose	39
6.3.2 Structure	39
6.3.3 Parameters	40
6.3.4 Service procedure	41
6.3.5 Conformance	41
7 Data set management services	42
7.1 Data set description	42
7.1.1 The data set object	42
7.1.2 Data set contents	44
7.1.3 Operations on the data set	44
7.2 InitiateLoad service	45
7.2.1 Purpose	45
7.2.2 Structure	45
7.2.3 Parameters	45
7.2.4 Service procedure	46
7.3 LoadSegment service	47
7.3.1 Purpose	47
7.3.2 Structure	47
7.3.3 Parameters	47
7.3.4 Service procedure	48
7.4 TerminateLoad service	48
7.4.1 Purpose	48
7.4.2 Structure	49
7.4.3 Parameters	49
7.4.4 Service procedure	49
7.5 InitiateUpLoad service	49
7.5.1 Purpose	49
7.5.2 Structure	50
7.5.3 Parameters	50
7.5.4 Service procedure	50
7.6 UpLoadSegment service	51
7.6.1 Purpose	51
7.6.2 Structure	52
7.6.3 Parameters	52
7.6.4 Service procedure	53
7.7 TerminateUpLoad service	53
7.7.1 Purpose	53
7.7.2 Structure	53
7.7.3 Parameters	53
7.7.4 Service procedure	53
7.8 GetDataSetAttribute service	54
7.8.1 Purpose	54
7.8.2 Structure	54
7.8.3 Parameters	54
7.8.4 Service procedure	55

8 VAA management service.....	56
8.1 VAA description	56
8.1.1 Purpose.....	56
8.1.2 The VAA object.....	57
8.1.3 Operations on the VAAs	58
8.2 ChangesScope service	58
8.2.1 Purpose.....	58
8.2.2 Structure	58
8.2.3 Parameters.....	58
8.2.4 Service procedure.....	59
9 Task invocation management services	60
9.1 Task invocation description	60
9.1.1 The TI object.....	60
9.1.2 Operations on the TIs	62
9.2 Start service.....	62
9.2.1 Purpose.....	62
9.2.2 Structure	62
9.2.3 Parameters.....	62
9.2.4 Service procedure.....	63
9.3 Stop service	64
9.3.1 Purpose.....	64
9.3.2 Structure	64
9.3.3 Parameters.....	64
9.3.4 Service procedure.....	65
9.4 Resume service.....	65
9.4.1 Purpose.....	65
9.4.2 Structure	66
9.4.3 Parameters.....	66
9.4.4 Service procedure.....	66
9.5 MakeUsable service	67
9.5.1 Purpose.....	67
9.5.2 Structure	67
9.5.3 Parameters.....	67
9.5.4 Service procedure.....	68
9.6 GetTIAttribute service	69
9.6.1 Purpose.....	69
9.6.2 Structure	69
9.6.3 Parameters.....	69
9.6.4 Service procedure.....	70
10 Variable access services	71
10.1 Variable model description	71
10.1.1 Variable Access model.....	71
10.1.2 The Named Variable object	73
10.1.3 The Message Box object	75
10.1.4 The Named Variable List object	77
10.2 Specification of types	78
10.2.1 Type Description parameter	79
10.2.2 Detailed Access parameter	80
10.3 Specification of data values.....	83
10.3.1 Data parameter	83
10.3.2 Data Access Error parameter.....	84
10.3.3 Variable Access Specification parameter	85
10.4 Read service	86
10.4.1 Purpose.....	86
10.4.2 Structure	86
10.4.3 Parameters.....	86
10.4.4 Service procedure.....	87

10.5 Write service	87
10.5.1 Purpose.....	87
10.5.2 Structure.....	88
10.5.3 Parameters.....	88
10.5.4 Service procedure.....	89
10.6 UnconfirmedWrite service	89
10.6.1 Purpose.....	89
10.6.2 Structure.....	89
10.6.3 Parameters.....	90
10.6.4 Service procedure.....	90
10.7. InformationReport service.....	90
10.7.1 Purpose.....	90
10.7.2 Structure.....	91
10.7.3 Parameters.....	91
10.7.4 Service procedure.....	92
10.8 GetVariableAttribute service.....	92
10.8.1 Purpose.....	92
10.8.2 Structure.....	92
10.8.3 Parameters.....	93
10.8.4 Service procedure.....	94
Figures.....	17 to 22
Annexes	
A DLMS protocol	95
A.1 Conventions.....	95
A.2 Useful types.....	97
A.3 DLMS PDUS	98
A.4 Service requests and responses.....	100
A.5 Service error	102
A.6. Object name	103
A.7 Context management protocol.....	104
A.8 VDE support protocol	105
A.9 Data set management protocol	106
A.10 VAA management protocol.....	107
A.11 Task invocation management protocol.....	108
A.12 Variable access protocol.....	109
B DLMS user states description.....	113
B.1 Overview	113
B.2 Context management states	114
B.3 Data set management states	116
B.4 Task invocation management states.....	122
ZA (normative) Normative references to international publications with their corresponding European publications.....	125

INTRODUCTION

This specification defines the distribution line message specification (DLMS) within the OSI application layer in terms of:

- a) an abstract model defining the interaction between users of services;
- b) the externally visible functionality of implementations conforming to this specification, in the form of procedural requirements associated with the execution of service requests;
- c) the primitive actions and events of services;
- d) the parameter data associated with each primitive actions and events;
- e) the relationship between, and the valid sequences of, these actions and events.

The services defined in this specification are provided by the distribution line message specification protocol. They may be used by other application layer service elements or by other elements of the application process.

This specification does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system. This specification specifies the externally visible functionality of implementations and the conformance requirements for such functionality.

**DISTRIBUTION AUTOMATION USING
DISTRIBUTION LINE CARRIER SYSTEMS –**

**Part 4 : Data communication protocols –
Section 41: Application protocols –
Distribution line message specification**

1 Scope

The scope of application of the specifications of the sections of part 4 is the communication through the so-called distribution line carrier technology (DLC) on both low and medium voltage distribution network. The application range based on telecommunication processes is wide and cannot be described exhaustively in this section; application examples are: control and monitoring of the distribution network, broadcasting of orders, control of user interfaces, public lighting, traffic lights supervision, automatic meter reading, etc.

Extensions to other communication media are also allowed.

The distribution line message specification (DLMS) is an application layer specification designed to support messaging communications to and from distribution devices in a computer integrated environment. This environment is referred in this specification as the distribution environment. This specification does not specify a complete set of services for remote programming of devices.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this section of IEC 1334-4. At the time of publication, the editions indicated were valid. All normative documents are subject to revision, and parties to agreements based on this section of IEC 1334-4 are encouraged to investigate the possibility of applying the most recent editions of the normative documents listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

IEC/FDIS 1334-4-42: *Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 42: Application protocols – Application layer*¹

ISO 7498: 1984, *Information processing systems - Open Systems Interconnection - Basic Reference Model*

ISO/IEC 7498-3: 1989, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 3 : Naming and addressing*

ISO/IEC/TR 8509: 1987, *Information processing systems - Open Systems Interconnection - Service conventions*

ISO/IEC 8649: 1988, *Information processing systems - Open Systems Interconnection - Service definition for the Association Control Service Element*

ISO/IEC 8650: 1988, *Information processing systems - Open Systems Interconnection - Protocol specification for the Association Control Service Element*

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

ISO/IEC 8825: 1990, *Information technology - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*

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

¹ At present at the stage of Final Draft International Standard (57/265/FDIS).

3 Definitions

3.1 Reference model definitions

This specification is based on the concepts developed in the Basic Reference Model for Open Systems Interconnection (ISO 7498).

It makes use of the following terms defined in that International Standard:

- a) **application-entity (AE);**
- b) **application-process (AP);**
- c) **application service element (ASE);**
- d) **open system;**
- e) **(N)-protocol;**
- f) **(N)-protocol-data-unit (N-PDU);**
- g) **(N)-service-access-point (N-SAP);**
- h) **(N)-layer;**
- i) **system;**
- j) **(N)-user-data.**

3.2 Service convention definitions

This specification makes use of the following terms defined in ISO/IEC/TR 8509 as they apply to the distribution line message specification:

- a) **primitive;**
- b) **request;**
- c) **indication;**
- d) **response;**
- e) **confirm;**

- f) **service primitive**;
- g) **service provider**;
- h) **service user**.

3.3 Other definitions

For the purpose of this specification, the following definitions also apply:

- 3.3.1 **attribute**: A data element, having a defined meaning, together with a statement of the set of possible values it may take.
- 3.3.2 **data**: Any representation to which meaning is or might be assigned (for example characters).
- 3.3.3 **data set**: An abstract object that represents a subset of the capabilities of a VDE which is used for a specific purpose.
- 3.3.4 **Distribution Line Protocol Machine (DLPM)**: An abstract machine that carries out the procedures specified in this specification.
- 3.3.5 **DLMS context**: A specification of the service elements of DLMS and semantics of communication to be used during the lifetime of an application association.
- 3.3.6 **Key attribute** : Attribute of an object type which is part of a combination of attributes that uniquely identifies the object.
- 3.3.7 **loading**: The process of transferring the contents of a data set from a DLMS client to a DLMS server.
- 3.3.8 **predefined object**: An object that exists at the VDE and that may not be created with DLMS services.
- 3.3.9 **standardized object**: An instance of an object which scope of access is VDE-specific or VAA-specific and the definition of which is provided in this specification.
- 3.3.10 **type**: An abstract description of the data class which may be conveyed by the value of a variable.
- 3.3.11 **uploading**: The process of transferring the contents of a data set (or part of them) from a DLMS server to a DLMS client.
- 3.3.12 **VAA-specific (Virtual Application Association specific)**: An adjective used to describe an object which may be accessed only by the application association which has previously created the VAA object.
- 3.3.13 **variable**: One or more data elements that are referred all together by a single name.
- 3.3.14 **variable access**: The access mechanism to variables or variable components defined at the VDE.
- 3.3.15 **VDE-specific**: An adjective used to describe an object which may be accessed by all the application associations established with the VDE.

3.4 Abbreviations

The following abbreviations are used in DLMS:

(A-1)SAP	: Underlying application service access point with (A-1) representing the layer below the application layer.
AA	: Application association
ACSE	: Association control service element
AE	: Application entity
AP	: Application process
ASE	: Application service element
ASN.1	: Abstract syntax notation one
DLMS	: Distribution line message specification
DLPM	: Distribution line protocol machine
DS	: Data set
OSI	: Open systems interconnection
PDU	: Protocol data unit
SAP	: Service access point
SDU	: Service data unit
TI	: Task invocation
VAA	: Virtual application association
VDE	: Virtual distribution equipment

3.5 Conventions

3.5.1 Service parameter description

This specification uses a tabular format to describe the component parameters of the DLMS service primitives. Each table consists of three to five columns describing the service parameters and the request ("Req"), indication ("Ind"), response ("Resp"), and confirm ("Conf") primitives. The "Resp" and "Conf" columns are absent when the service is not a confirmed service.

In each table, one parameter (or a part of it) is listed on each horizontal line. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter.

The codes used are listed below:

M - the parameter is mandatory for the primitive;

U - the parameter is a user option, and may or may not be provided depending on dynamic usage by the DLMS user;

S - the parameter is selected among other S-parameters as internal response of the DLMS server environment;

C - the parameter is conditional upon other parameters or the environment of the DLMS user;

(blank) - the parameter is never present.

The "(=)" code following one of the M, U, C or S codes indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table. (For instance, an "M(=)" code in the indication service primitive column and an "M" in the request service primitive column means that the parameter in the indication primitive is semantically equivalent to the one in the request primitive).

Some parameters may contain subparameters. Subparameters are indicated by labelling of the parameters as M, U or C, and indenting all subparameters under the parameter. Presence of the subparameters is always dependent on the presence of the parameter that they appear under (for example, an optional parameter may have subparameters; if the parameter is not supplied, then no subparameters may be supplied).

The descriptions of parameters in this specification make reference to types, in order to describe the allowable values for such parameters. The types referenced are defined in ISO/IEC 8824.

3.5.2 Addressing in DLMS

The DLMS specification does not provide the means for naming and addressing of a peer DLMS user or peer DLPM. Within OSI and within DCP, the identification and addressing of peer application entities is carried out through the use of ACSE services defined in ISO 8649. After such an association between peers has succeeded, all of the DLMS PDUs flow between these peers over the (A-1)-LAYER services. It is therefore not necessary for DLMS to carry addressing information. Additional information on naming and addressing may be found in the application layer specification.

3.5.3 Service conventions

This specification uses the descriptive conventions contained in the OSI Service Conventions (ISO/IEC/TR 8509). The OSI service conventions define the interactions between the DLMS user and the DLMS provider. Information is passed between the DLMS user and the DLMS provider by service primitives, which may convey parameters.

The following remarks apply to the use of this model:

- a) ISO/IEC/TR 8509 defines a model for the service provided by a layer of the OSI reference model. The DLMS service does not correspond to such a layer (it describes a part of the application layer) but the model used is identical in all other respects;
- b) at any instant in time, an AE has multiple service requests outstanding, each proceeding independently of the others.

NOTE – It should be noted that the DLMS user/DLMS provider distinction is an abstraction, and may not necessarily correspond to a realization of DLMS in any particular system.

3.5.4 Sending and receiving DLMS user and DLPM

This specification makes use of the terms sending and receiving DLMS users. The sending DLMS user is the DLMS user that issues a request or response service primitive. The receiving DLMS user is the DLMS user that receives an indication or confirmation service primitive.

NOTE – It is important to note that, in the course of completion of a confirmed DLMS service, both DLMS users will be sender and receiver at the same time. The first DLMS user sends the request and receives the confirmation, while the second DLMS user receives the indication and sends the response.

This specification makes use of the terms sending and receiving DLPMs. The sending DLPM is the DLPM that sends a DLMS PDU. The receiving DLPM is the DLPM that receives a DLMS PDU.

3.5.5 Requesting and responding DLMS user

This specification makes use of the terms requesting and responding DLMS users. The requesting DLMS user is the DLMS user that issues the request service primitive for a service, while the responding DLMS user is the DLMS user that issues the response service primitive for a service.

NOTE – It is important to note that the use of the term responding DLMS user differs from the use of the term responding entity in ACSE and other standards. In those standards, the term is used to reference the entity that responds to a connection request.

3.5.6 Client and server of a service

This specification makes use of the terms client and server in order to describe the model of the DLMS VDE. The server is defined as the system that behaves externally as a VDE for a particular service request instance. The client is the system that makes use of the VDE for some particular purpose via a service request instance. The VDE is primarily useful in describing the actions of the server, and thus is describing the requests and responses that a client may make use of.

3.5.7 Object modelling

This specification makes use of a technique of abstract object modelling in order to fully describe the DLMS device model and the DLMS service procedures. In this modelling technique, abstract objects with characteristics and possible operations are described. The objects defined are abstract, and aid in the understanding of the intent of DLMS service procedures and their effects. In implementing DLMS, a real system maps the concepts described in the model to the real device. Hence, as viewed externally, a device that conforms to this specification exhibits the characteristics described in the object modelling technique, but the mechanisms for the realization of this view are not defined by this specification.

The DLMS model defines a set of objects. Each object constitutes an abstract entity which exhibits certain characteristics and may be affected by certain DLMS services and operations. Each object type is given a name, by which it may be referenced.

Each object type is characterized by a set of attributes, which serve to describe some externally visible features of the object. Some DLMS services may modify the characteristics of an object instance, and hence their effect on the device may be modelled by a change in one or more attributes of an object.

Each object instance within an object type must be uniquely identified among all objects. For this purpose, one of the objects attributes must be unique (for example, many objects have a name attribute, which is different for each object instance of the object type). In DLMS, each attribute which makes the object unique is identified as a "Key attribute".

Finally, some objects contain attributes which are conditional, in the sense that they are relevant to the object if and only if certain conditions are true. DLMS expresses such attributes through the use of "constraint", which specify a condition. Attributes that are subject to a constraint are considered as object attributes for an object instance if, and only if, the constraint is satisfied for that object instance.

In DLMS, object types are syntactically defined as follows:

Object: (name of object)
 Key attribute: (name of attribute)
 attribute: (name of attribute)
 .
 .
 .
 constraint: (constraint expression)
 attribute: (name of attribute)
 attribute: (name of attribute)
 attribute: (name of attribute)

By convention, each object definition begins with an object declaration and the name of the object. Immediately following and indented, one Key attribute is named. Next, zero, one or more attributes are named. Constraints may be expressed anywhere within the attributes, with the convention that all the constrained attributes are indented underneath it. The first attribute definition that is not indented ends the list of attributes that are subject to the constraint.

4 Virtual distribution equipment

4.1 Relationship of the VDE to the OSI model

4.1.1 The VDE within the AP

A VDE exists within the DLMS AP. It constitutes that portion of an information processing task which makes available, for transmission of meaningful information, a set of resources and functionalities associated with a real distribution device.

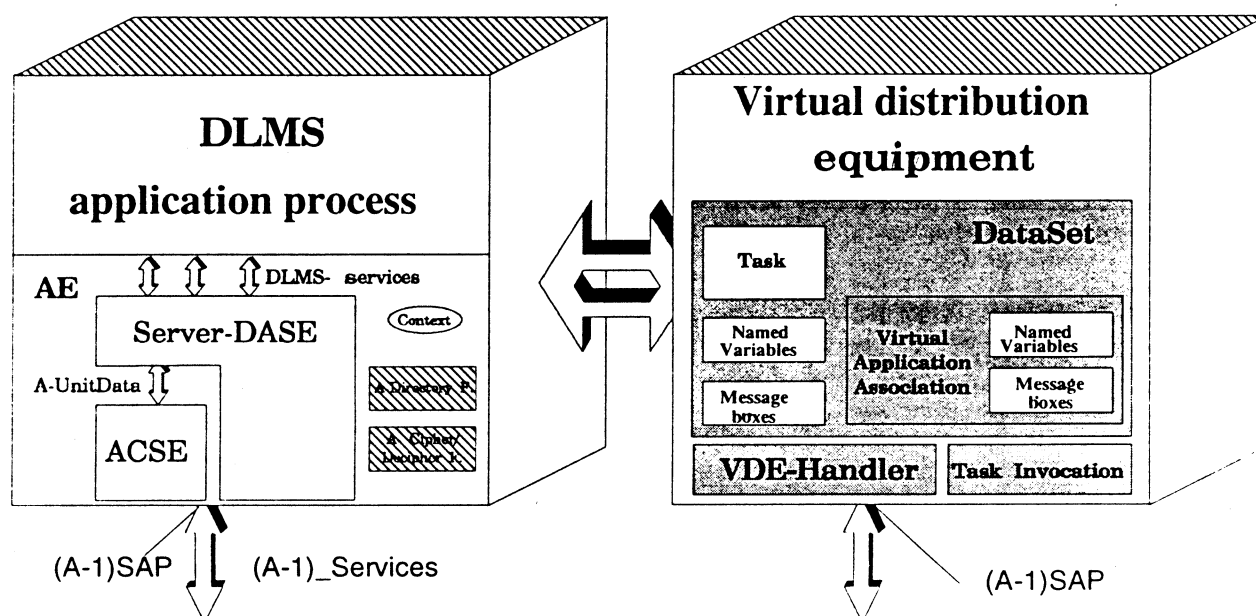


Figure 1 - Distribution equipment model

An AP may define zero, one or more virtual distribution equipments. If it does not define at least one VDE it may not act as a DLMS server.

Each VDE represents a virtual distribution device within an AP that models a coherent part of the AP behaviour. Each VDE is logically separated from all the other VDEs.

Example:

A DLMS system which is connected to a non-DLMS environment containing multiple attached distribution devices like meters, switches or control units could be modelled:

- as a single AP containing one VDE for each attached device;
- as several application processes, each containing a single, distinct VDE for each attached device.

The clients of the VDEs in each case will see a particular attached device as a single VDE. Relative to the DLMS services, this VDE appears to be independent of all the other VDEs.

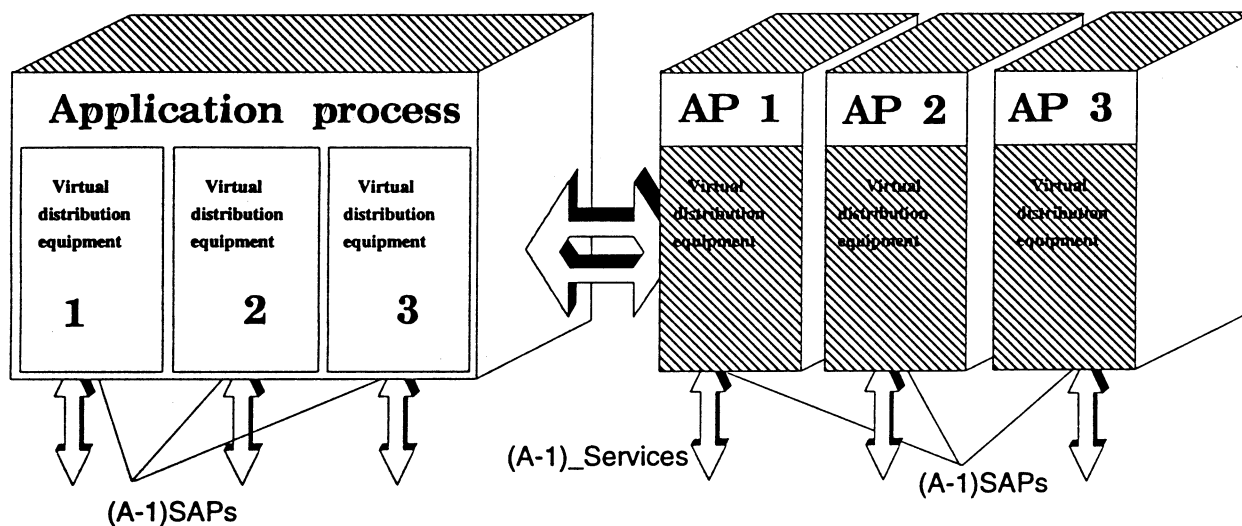


Figure 2 - VDE within APs

According to the equivalence between several APs containing each a single VDE and a single AP containing several VDEs, it is assumed in the following that an AP contains only one VDE.

4.1.2 The AE within the VDE

As described in ISO 7498, an AE represents a set of communication capabilities of the AP. An AE models the aspects of an AP which need to be taken into account for the purpose of OSI communications. An AP without AE cannot communicate in the OSI environment or in the DCP environment. Please refer to the DCP application layer specification or to the ISO 7498, ISO/IEC 9545 and ISO/IEC 8649 standards for further details.

Each VDE may contain one or more AEs. Each AE represents the set of communication capabilities of the AP part that is modelled with the VDE. A VDE that contains no AE cannot communicate in the DCP environment. Each AE is a part of one and only one VDE. Communication with a particular AE is used to model communication with a VDE, and hence with an AP.

NOTES

1 DLMS assumes that there is only one AE of a specified type, corresponding to a particular (A-1)SAP in an AP and thus in a VDE. If an AE of another AE-type exists within the AP, it shall use another (A-1)SAP address. Such a configuration is not further taken into account.

2 From the underlying layer, each AE is identified by a single (A-1)SAP address and may be therefore consistently addressed within OSI.

An AA is a cooperative relationship between two AEs. It provides the necessary frame of reference between the AEs in order that they may interwork effectively.

The AE-invocation represents a specific use of the capabilities of an AE in an instance of communication. Each AE-invocation models an instance of usage of that AE in an AA. Several AE-invocations of an AE are used to represent several application associations.

NOTE 3 – DLMS does not allow multiple application associations in a single AE-invocation. Multiple AAs are modelled as multiple AE-invocations.

A given VDE is addressed through an AE by a single (A-1)SAP. This binding of an (A-1)SAP to a VDE is of long duration. The equivalence between an AP acting as a DLMS server and its VDE is shown in figure 3.

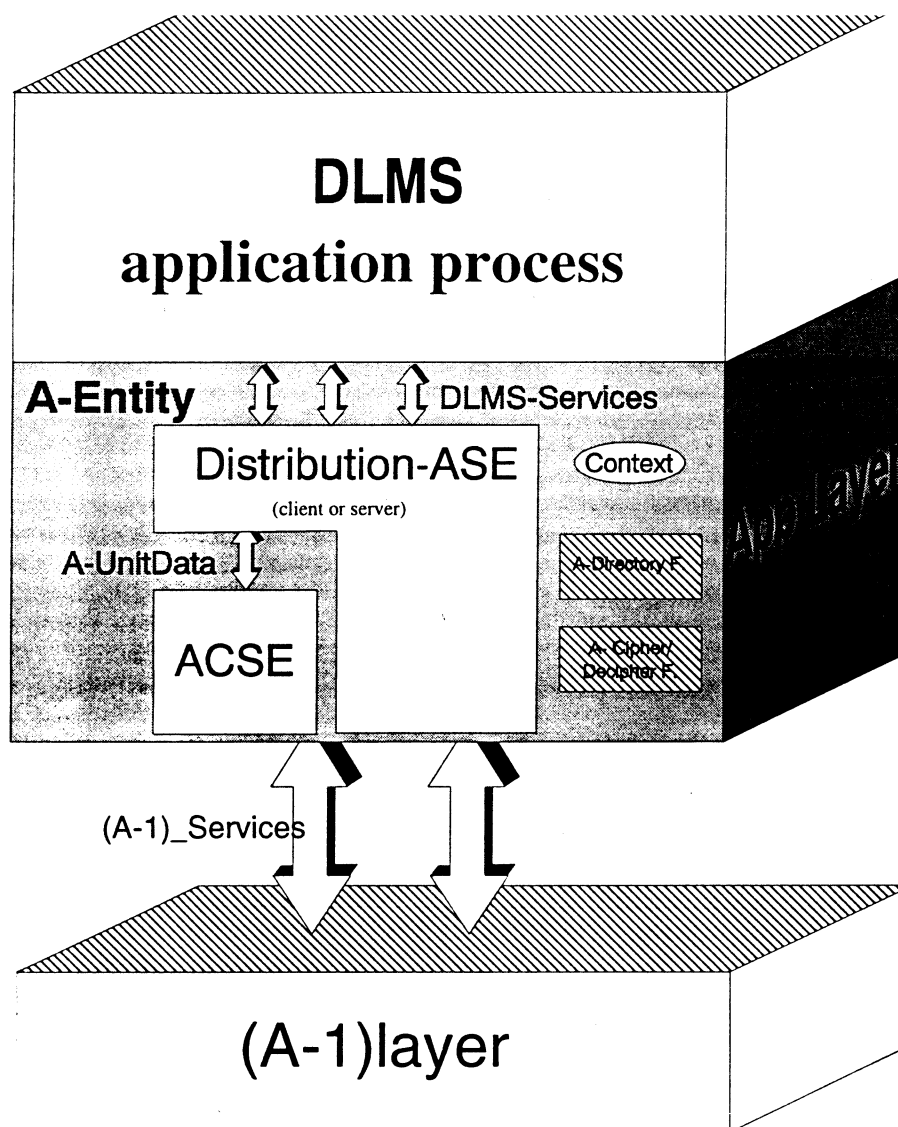


Figure 3 - Application process structure

4.2 Relationship with a real distribution device

A VDE is an abstract representation of a specific set of resources and functionalities of a real distribution device and a mapping of this abstract representation to the physical and functional aspects of the real distribution device. This mapping of a virtual resource to the underlying real resource is of long duration (comparable in fact to the lifetime of the hardware).

Generally, the resources of a given VDE are distinct from, and independent of, the resources of all the other VDEs. When virtual resources of several VDEs are mapped to the same underlying physical resource, a mechanism shall be provided by the application processes to co-ordinate their accesses to the single resource. A semaphore usually controls access to the real shared resource. Semaphores are not further discussed in DLMS. The manipulation of real semaphores is considered to be a local issue.

NOTE – A "named variable" object can be used to represent the states of a semaphore within a VDE.

DLMS describes the operation of a VDE by describing the abstract objects which are manipulated by it. It also describes the set of operations which may be performed on these objects through the use of DLMS services.

4.3 Structure of a VDE

4.3.1 Structure overview

Each VDE contains precisely:

- one VDE-handler;
- one data set;
- zero, one or more TIs;
- zero, one or more variables;

The following figure 4 shows the constituent parts of a VDE.

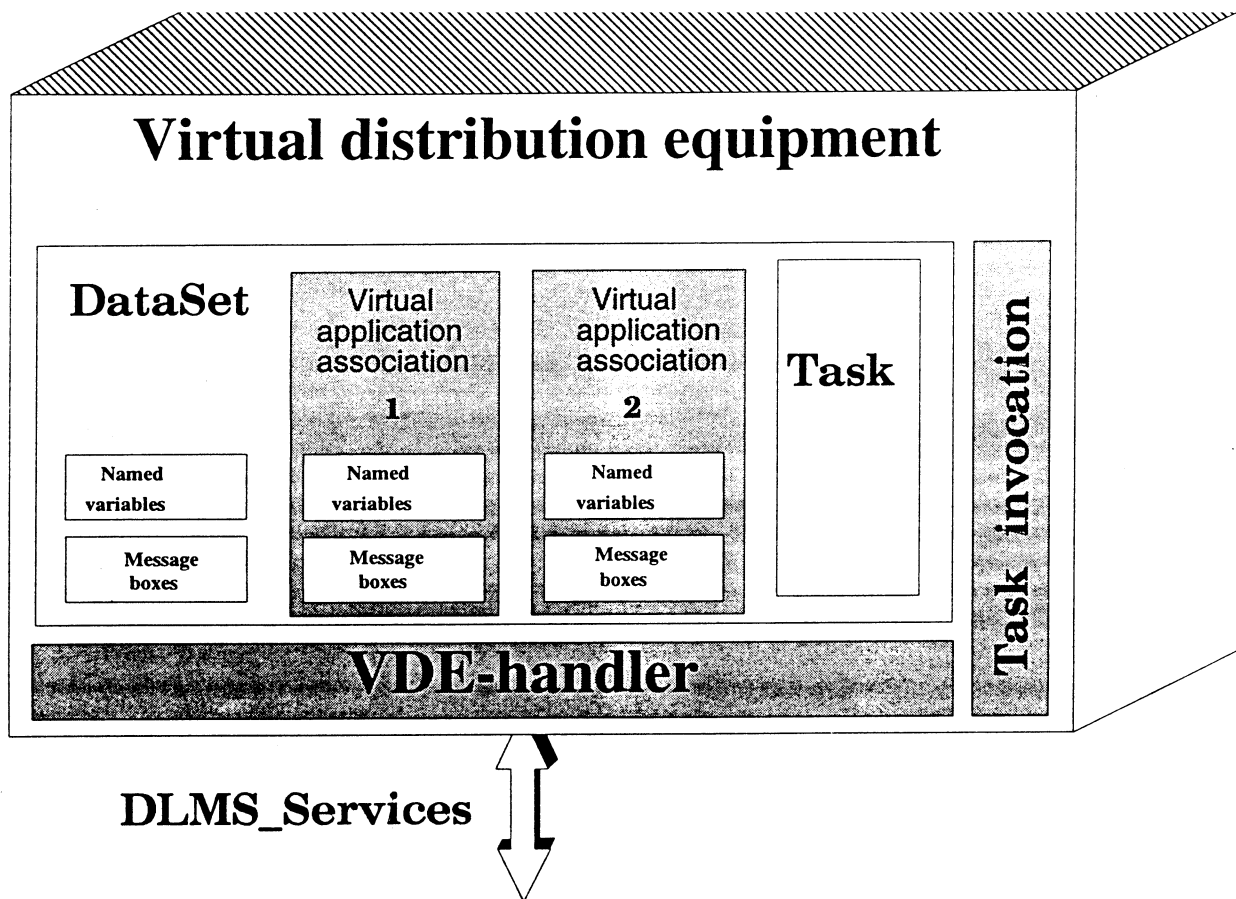


Figure 4 - The VDE structure

4.3.2 The VDE-handler

The VDE-handler manages all access using the DLMS services to the built-in resources of a VDE. It represents the interface between the DCP environment and the local operating environment of the VDE. Conceptually, the VDE-handler includes all aspects of the local operation of the VDE which are visible to its clients, except for those aspects which are explicitly defined by the data set. Thus, the VDE-handler supports DLMS visibility to the local operating environment as well as visibility to locally defined procedures which may be invoked using DLMS.

The VDE-handler contains the resources of the VDE. Some of the resources are used by the data set

The VDE-handler represents the built-in functions of the VDE which are necessary for it to act as a VDE. The existence of a fully functional VDE-handler corresponds exactly with the existence of the VDE.

4.3.3 The data set

In DLMS, the data set object represents the set of information that is bound to a specific purpose of the VDE. The data set may be empty or may contain information. The information may be program instructions, values tables, various lists of named variables or special messages. The data set, as a set of information, may be loaded using DLMS services.

More conceptually, the data set represents the set of VDE resources which is used for achieving a specific purpose of the VDE. The data set includes those aspects of the VDE which are associated with a co-ordinated distribution strategy.

The allocation of the VDE resources to its data set is static. The data set resources are predefined within the DLMS server and cannot be changed using DLMS services.

Variable objects may be defined subordinated to the data set, which then represents a single name space for those DLMS variables.

An AP can load the contents of the data set by using the data set management services.

4.3.4 The task invocation

The TI is an element which most closely corresponds to putting a program on the execution queue in a multi-tasking environment. A TI is associated with a task that contains the code to be executed and the associated variables. A TI contains all the information necessary to initiate the execution of the corresponding task within the VDE.

In DLMS, the TI is one of the predefined objects at the VDE. The corresponding task itself may be loaded with a data set content. Further details on tasks may be found in other sections of IEC 1334.

An AP can manipulate the TIs by using the task invocation management services.

4.3.5 The variable

The variable is an abstract element of the VDE which is capable of providing (when read) or accepting (when written) a typed data value. A data type or simply type is an abstract description of the class of data which may be conveyed by a variable value. A variable type determines its abstract syntax, its range of possible values, and its representation while being communicated using DLMS.

4.4 Specification of DLMS objects

This subclause defines some basic attributes of the DLMS objects.

4.4.1 Object name

DLMS objects are referenced by name. In DLMS, access to an object is only possible through the use of its name. Actually, there is no means to make a direct raw access to a real resource within DLMS. Object names are intended to be the Key attributes of the DLMS objects. The name of an object shall then be unique within its space of definition. This space is known as the scope of name of the object.

4.4.2 Scope of name

In DLMS, only one scope of name is defined: VDE width. The scope of name is then the same for all the objects in a VDE. An object name shall be uniquely defined within the VDE regardless to its object class. An object name may be referenced by every client of the VDE.

4.4.3 Scope of access

The scope of access defines the restriction of accessibility to an object of a VDE instance. This is to avoid non-authorized manipulation and confidentiality of data in a VDE.

In DLMS, two scopes of access are defined: VDE-specific and VAA-specific.

An object with a VDE-specific scope of access may be manipulated by all the peer DLMS users of that VDE. For example a named variable with a VDE-specific scope is freely accessible. Every peer DLMS user may attempt to read the variable.

A VAA-specific scope of access is directly bound to the existence of a corresponding VAA object. The VAA-specific scope defines a virtual space of access that is open to the DLMS user which has previously created the VAA object and is closed to all the other DLMS users.

The scope of access of a DLMS object may be changed through the use of the ChangeScope service described in clause 8.

4.4.4 Lifetime

The lifetime of an object expresses whether an object is defined subordinated to the VDE or to the data set. In DLMS, the structure of the VDE is static; an object defined subordinated to the VDE exists as long as the VDE exists. Its lifetime is then the same as the VDE. The data set may be loaded using DLMS services. The nature of the data set is then more dynamic. An object defined subordinated to the data set exists as long as the data set exists. Its lifetime is then the same as the data set one. At a VDE, an object subordinated to the data set may then appear or disappear when the data set is loaded again.

The lifetime of a DLMS object is specified in the lifetime attribute.

4.4.5 Object classes

The specific instances of DLMS objects which can be named are listed in the following table. The allowed combinations are indicated in association with the inferred lifetime of the objects.

Table 1 - Object classes

Object class	VDE-specific	VAA-specific	Lifetime
Named Variable objects	x	x	DS/VDE
Named Variable List objects	x	x	DS/VDE
Message Box objects	x	x	DS/VDE
TI objects	x	x*	VDE
VAA objects	x		VDE
Data Set object	x	x*	VDE
* If the scope of access of the data set is VAA-specific, then all the TI objects have the same VAA-specific scope. If the scope of access of a TI object is VAA-specific, then all the other TI objects and the data set have the same VAA-specific scope. The associated VAA object is known to be the Executive VAA.			

Each of the objects listed above has a lifetime within the VDE which can be inferred from its type. The VDE lifetime objects exist as long as the VDE exists unless explicitly changed. For example, the data set object has a VDE lifetime because it exists in the VDE as long as the VDE exists unless explicitly changed with the data set management services. The data set lifetime objects are contained within the data set and exist as long as the data set exists.

4.4.6 Object Name parameter

The parameter Object Name occurs frequently in the specification of DLMS services. The Object Name parameter identifies uniquely within a VDE a DLMS object. The Object Name contains in its structure the class of the object. This makes VDE-handler operations easier.

The structure of the Object Name parameter is described as follow:

Object name

attribute: object class (NAMED-VARIABLE, NAMED-VARIABLE-LIST, MESSAGE-BOX, TI, DATA-SET, VAA)

attribute: Item Identifier

The Object Class parameter identifies the class to which the object belongs. It is one of the DLMS defined object class. The Item Identifier parameter, identifies uniquely the object name within the VDE for a particular object class.

The Object Name is always referenced as a whole. The semantic construction of the object name is provided only to ensure comprehensive manipulation of the DLMS objects. It shall be clear that reference to an object using its name provides also identification of its class. The ASN.1 description of the Object Name parameter is given in annex A as an Integer16.

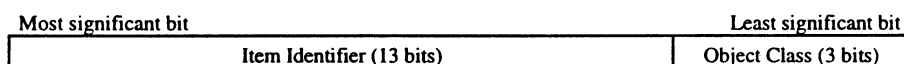


Figure 5 - Object name structure

Object class values are chosen as follows :

- 000 Named Variable object;
- 001 Named Variable List object;
- 010 Message Box object;
- 011 TI object;
- 100 Data Set object;
- 101 reserved;
- 110 reserved;
- 111 VAA object.

The Item identifier values are described on 13 bits. Values may be freely chosen.

4.4.7 DLMS object description

Most of the DLMS object descriptions begin as follows:

Class of object

key attribute: Object Name

attribute: Scope Of Access (VDE-specific, VAA-specific)

constraint Scope Of Access = VAA-specific

attribute: VAA Name

attribute: Scope May Change (TRUE, FALSE)

attribute: Lifetime (VDE, DATA-SET)

attribute: ...

The key attribute is always the object name. It is provided to uniquely identify the DLMS object among all other objects defined at the VDE and regardless of its class.

The Scope Of Access parameter defines the current scope of access for the designated object. This scope of access rules the object access. It shall be VDE-specific or VAA-specific. If the Scope Of Access attribute contains the VAA-specific value, the name of the VAA is specified in the VAA Name attribute.

The Scope May Change attribute defines whether or not the scope of access of the related object may change using the DLMS ChangeScope service.

The lifetime attribute specifies which kind of lifetime the described object has. It shall be VDE or DATA-SET. A lifetime attribute set to DATA-SET indicates that the object is defined within the data set.

Other attributes are more specific to the object class.

Alteration of DLMS objects : All DLMS objects have as part of their description a list of externally visible attributes. In addition to the DLMS services which alter these attributes, they may also be changed through the local action of the system or of the system operators, or through the execution of a task invocation. The means of local alteration of Object Attributes is outside the scope of this standard.

4.5 Conformance description**4.5.1 Purpose**

The Conformance Block is proposed in order to allow various real environments to communicate using the same DLMS protocol through different capabilities. In DLMS, it is not mandatory to implement all the described services. Only the Initiate, Abort, GetStatus and a simple GetNameList services are mandatory. Other services may be negotiated using the Conformance Block. An implemented service shall be fully conform to its DLMS specification. A non-mandatory service described in the Conformance Block with a non-set bit may not be requested by a DLMS client.

4.5.2 Structure**Table 2 - Conformance description**

Conformance description	Code
Conformance	M
GetDataSetAttribute	M
GetTIAtribute	M
GetVariableAttribute	M
Read	M
Write	M
UnconfirmedWrite	M
ChangeScope	M
Start	M
Stop Resume	M
MakeUsable	M
DataSet Load	M
Selection in GetNameList	M
Detailed Access	M
Multiple Variable List	M
Data Set Upload	M

4.5.3 Parameters

The GetDataSetAttribute flag describes whether or not the GetDataSetAttribute service is supported.

The GetTIAtribute flag describes whether or not at least one TI object is defined at the VDE. If at least one TI object exists, the GetTIAtribute service shall be supported, fully implemented and fully conform to its description in this standard. No partial conformance is allowed for this service.

The `GetVariableAttribute` flag describes whether or not at least one variable object (a `Named Variable` object, a `Named Variable List` object or a `Message Box` object) is defined at the VDE. If at least one variable object exists, the `GetVariableAttribute` service shall be supported, fully implemented and fully conform to its description in this present document. No partial conformance is allowed for this service.

The `Read`, `Write`, `UnconfirmedWrite`, `ChangeScope`, `Start` and `MakeUsable` flags describe whether or not the corresponding services are available at the VDE. If a service is described as available, it shall be fully implemented and fully conform to its description in this standard. No partial conformance is allowed for these services.

The `Stop Resume` flag describes whether or not `Stop` service and `Resume` service are available at the VDE. If this flag is set, both services shall be fully implemented and fully conform to their descriptions in this standard. No partial conformance is allowed for these services.

The `DataSet Load` flag specifies whether `InitiateLoad` service, `LoadSegment` service and `TerminateLoad` service are available at the VDE. If this flag is set, all three services shall be fully implemented and fully conform to their descriptions in this standard. No partial conformance is allowed for these services.

The `Selection in GetNameList` flag describes whether selection in the `GetNameList` service is supported. `GetNameList` service in its simplest presentation shall always be supported. That is a `GetNameList` where no special selection is requested by the client. The `GetNameList` service description introduces also a more sophisticated way to list object names defined at a VDE. It is possible to select names by their lifetime, by their object class or by their scope of access. This enhancement of the basic `GetNameList` service shall be supported if the `Selection in GetNameList` flag is set.

The `Detailed Access` descriptor specifies on two bits the supported nested level for detailed accesses. The description is as follows:

- 00: No Detailed Access is supported;
- 01: Detailed Access is supported for one single level;
- 10: Detailed Access is supported for two levels;
- 11: Detailed Access is fully supported, whatever may be the recursivity.

The `Multiple Variable List` flag indicates whether the implementation is capable of handling, in `Read` and `Write` indications and in `InformationReport` requests, lists of variables that are composed of more than one variable. If not set, the server may only be able to process lists consisting of one single variable of class `Named Variable` or `Message Box`.

The `Data Set Upload` flag specifies whether `InitiateUpload` service, `UploadSegment` service and `TerminateUpload` service are available at the VDE. If this flag is set, all three services shall be fully implemented and fully conform to their descriptions in this standard. No partial conformance is allowed for these services.

5 Context management services

5.1 Introduction

The context management services are the Initiate service and the Abort service.

These services allow the DLMS client:

- to declare its communication characteristics to a DLMS server in the DLMS context. This is to establish the requirements and resources that will support that communication;
- to abort a communication with a DLMS server in the DLMS context.

5.2 Initiate service

5.2.1 Purpose

The initiate service is used to establish the DLMS context. It allows the communicating DLMS users to exchange information about their resources and communication requirements.

The Initiate service provides also the means to identify the DLMS client in the DLMS environment through the creation of a VAA object. This VAA object is used to define the scope of access of the VDE objects related to this DLMS client.

The Initiate service must be performed successfully before any other services can be operated.

5.2.2 Structure

The structure of the component service primitives is shown below.

Table 3 - The Initiate service

Initiate service	Req	Ind	Resp	Conf
Argument	M	M(=)		
Dedicated Key	U	U(=)		
Response Allowed	U	U(=)		
Proposed Quality Of Service	U	U(=)		
Proposed DLMS Version Number	M	M(=)		
Proposed Conformance	M	M(=)		
Proposed Max Pdu Size	M	M(=)		
Result(+)			C	C(=)
Negotiated Quality Of Service			U	U(=)
Negotiated DLMS Version Number			M	M(=)
Negotiated Conformance			M	M(=)
Negotiated Max Pdu Size			M	M(=)
VAA Name			M	M(=)
Result(-)			C	C(=)
Error Type			M	M(=)
NOTES 1 The collection of these parameters may be contained in the user data portion of the A_Unit_Data request of the ACSE. In this case, additional parameters are required to complete the A_Unit_Data request. 2 The DLMS provider in the calling system may reduce the values of the parameters in the request primitive. The DLMS provider in the DLMS server may reduce the values of the parameters in the indication primitive. No other modification is permitted to these values. 3 The Result(+) or Result(-) primitives may be transmitted if and only if the value of the Response Allowed parameter is TRUE or this parameter is absent.				

5.2.3 Parameters

The Argument parameter conveys the specific parameters of the Initiate Service request.

The Dedicated Key parameter contains a ciphering key that may be used in subsequent transmissions to cipher DLMS PDUs exchanged between the same client and server. Its use is allowed only when the Initiate request PDU to which it belongs has been encoded using a Global Key.

NOTE – The ways of setting the Global Key or the Dedicated Key are outside the scope of this standard.

The Response Allowed parameter indicates that the Initiate service is confirmed (if TRUE) or unconfirmed (if FALSE). In the first case, a response by the DLMS server is mandatory. In the second case, it is forbidden.

NOTE – The intended purpose of this parameter is to make possible the broadcasting of Initiate requests.

The Proposed Quality Of Service parameter enables the client to request a specific grade of service from the server. The exact semantics of this parameter is outside the scope of this standard. It may be fixed by companion specifications or by an external agreement between the client and the server.

The Proposed DLMS Version Number parameter, of type Unsigned8, contains the highest DLMS version level supported by the DLMS client.

The Proposed Conformance parameter, of type BIT STRING, defines the list of the services and functionalities that are supported by the DLMS client. The purpose of this parameter is to allow the DLMS provider to allocate the appropriate resources to ensure communication. The exact content of the Conformance Block is described in 4.5.

The Proposed Max Pdu Size parameter, of type Unsigned16, proposes a maximum length expressed in bytes for the exchanged DLMS PDUs. The value proposed in an Initiate request must be large enough to always permit the Initiate Error PDU transmission.

The Result(+) parameter indicates that the requested service has succeeded.

The Negotiated Quality Of Service parameter enables the server to grant a specific grade of service to the client. The exact semantics of this parameter is outside of the scope of this standard. It may be fixed by companion specifications or by an external agreement between the client and the server.

The Negotiated DLMS Version Number parameter, of type Unsigned8, specifies the DLMS version level supported by the VDE-handler. The value of the parameter must be lower than or equal to the Proposed DLMS Version Number parameter.

The Negotiated Conformance parameter, of type BIT STRING, defines the list of the services and functionalities that may be used afterwards. This parameter is computed as a logical "and" between the submitted conformance and the VDE-handler internal conformance.

The Negotiated Max PDU Size parameter, of type Unsigned16, contains a maximum length expressed in bytes for the exchanged DLMS PDUs. A PDU that is longer than this maximum length will be discarded. This maximum length is computed as the minimum of the Proposed Max Pdu Size and the maximum PDU size that the VDE-handler may support.

The VAA Name parameter, of type ObjectName, identifies uniquely within the VDE, the VAA object associated with the DLMS client. Reference to this object may be made at any further time, especially to check the validity of an access to DLMS objects.

The Result(-) parameter indicates that the service previously requested has failed. The Error Type parameter provides the reason for failure. The Error Type parameter is described in details in A.5.

5.2.4 Service procedure

The Initiate service request may be issued at any time by the DLMS client.

On receipt of an Initiate indication service primitive, the DLMS server checks the provided parameters and the existence of an already existing associated VAA object. No response is issued if the Response Allowed parameter in the Initiate request primitive was present and set to FALSE.

A negative result is issued in an Initiate response primitive with the appropriate error code if one of the following statements is met:

- the proposed DLMS version number is lower than the supported one;
- the list of supported services is not sufficient;
- the maximum PDU size is not sufficient;
- a new VAA should be created but the VDE-handler cannot do it.

Otherwise, the response shall be successful. If the associated VAA object does not exist already, the VAA object is created. If it exists, the existing VAA object is reused.

NOTES

- 1 Companion specification may define conditions that are to be met to successfully perform an Initiate service.
- 2 If a PDU exceeds the negotiated PDU size, the PDU will be discarded; an Abort may be generated.
- 3 The VDE-handler shall have a deterministic behaviour. When receiving two Initiate indication primitives with the same parameters and referring to a common existing VAA, it shall return exactly the same response PDU.

5.3 Abort service

5.3.1 Purpose

The Abort service is used to relinquish the DLMS context abruptly, without negotiation. The DLMS client issues the Abort request primitive to indicate that it wishes to immediately, and without negotiation, discontinue a communication in the DLMS context.

5.3.2 Structure

Table 4 - The Abort service

Abort service	Req	Ind
Argument	M	M(=)

5.3.3 Parameter

The Argument parameter conveys a NULL value because no parameter is transmitted with the Abort service.

5.3.4 Service procedure

The Abort service request may be issued at any time by the DLMS client.

The DLMS server first checks the Abortable attribute of the associated VAA. If the Abortable attribute of the VAA is set to TRUE, the DLMS context is cancelled and the VAA object is deleted. If the Data Set was in the Loading state and within the scope of the VAA, then the state of this Data Set is set to Empty.

NOTE – This corresponds to deleting the fragments of the Data Set that may already have been loaded. All the objects defined with this VAA-specific scope of access have their scope changed to VDE-specific including both the Data Set and the TI objects if the VAA was the Executive VAA.

If the Abortable attribute of the VAA is set to FALSE, then nothing happens.

Before any further use of DLMS services, the DLMS client must perform again a successful Initiate service.

6 VDE support services

The Virtual Distribution Equipment is defined above. This clause describes the structure of the VDE object and the associated services. The VDE support services are the GetStatus and the GetNameList services.

6.1 Virtual distribution equipment description

6.1.1 The VDE object

The VDE represents the part of a DLMS AP whose behaviour is modelled as a single entity (see clause 4). The VDE is the main abstract entity of the DLMS specification.

6.1.1.1 Attributes

The attributes of the VDE are:

Object: VDE

key attribute: VDE-handler	(static)
attribute: VDE Type	(static)
attribute: Serial Number	(static)
attribute: Vendor Name	(static)
attribute: Model	(static)
attribute: Version Number	(static)
attribute: Resources	(static)
attribute: List of VAA	(dynamic)
attribute: Status	(dynamic)

The static attributes are pre-configured within a VDE. They are the VDE Type, the Serial Number, the Vendor Name, the Model, the Version Number and the Resources attributes. The static attributes are non-loadable and non-writable. They identify the vendor characteristics of the current implementation. The characteristics specified in these attributes are for revision management purposes.

The other attributes are dynamic. They describe the configuration of the VDE.

6.1.1.2 Description

The VDE-handler attribute identifies uniquely the VDE because the existence of a fully functional VDE-handler corresponds exactly with the existence of the VDE.

The VDE Type attribute specifies the type of the VDE. This parameter identifies uniquely the type of the VDE throughout the world. The attribution of a type to a VDE is out of the scope of this specification.

The Serial Number attribute identifies uniquely the VDE throughout the world.

The Vendor Name attribute identifies the manufacturer of the system which supports the VDE.

The Model attribute identifies the model of the system which supports the VDE.

The Version Number attribute identifies the software revision level of the communication system which supports the VDE.

The Status attribute indicates the logical status of the VDE. Three states are allowed: READY, NOCHANGE and INOPERABLE. The normal state of the VDE is the READY state, while the two other states are transient. In the READY state, all services are allowed. In the NOCHANGE state, the only services allowed are the services which do not modify the attributes of an object defined at the VDE, namely GetStatus, Read, InformationReport, GetNameList, GetDataSetAttributes, the three UpLoad services, GetTIAttribute and GetVariableAttribute. In the INOPERABLE state, none of the functions of the real device represented by the VDE is operational.

The Resources attribute describes the physical resources of the real distribution device represented by the VDE.

The List of VAA attribute identifies the VAA objects names that are currently defined within the VDE. The VAA object is described in clause 8.

6.1.2 VDE Content

From the DLMS point of view, the VDE contains zero or one Data Set, zero or more TIs and zero or more variables.

The variable object defined at the VDE may be Named Variable, Named Variable List or Message Box.

6.1.3 Operations on the VDE

The GetStatus and the GetNameList services operate on the VDE.

These services allow the DLMS client:

- to obtain the status of the VDE, its resources, the list of the defined VAA objects and, eventually, the specific characteristics of the manufactured system;
- to obtain the list of the objects defined at the VDE.

6.2 GetStatus service

6.2.1 Purpose

The purpose of the GetStatus service is to determine the general condition of a responding VDE. It may also identify the vendor specific attributes.

6.2.2 Structure

Table 5 - The GetStatus service

GetStatus service	Req	Ind	Resp	Conf
Argument	M	M(=)		
Identify	M	M(=)		
Result(+)			S	S(=)
VDE Type			M	M(=)
Serial Number			M	M(=)
Status			M	M(=)
List of VAA			M	M(=)
Identify			U	U(=)
Resources			C	C(=)
Vendor Name			C	C(=)
Model			C	C(=)
Version Number			C	C(=)
Result(-)			S	S(=)
Error Type			M	M(=)

6.2.3 Parameters

The Argument parameter conveys the specific parameter of the GetStatus service request.

The Identify parameter indicates to the responding user if it must include all the static attributes of the VDE in the GetStatus response primitive. If the parameter value is set to TRUE, the responding VDE indicates all its static characteristics in the GetStatus response primitive.

The Result(+) parameter indicates that the requested service has succeeded.