RAPPORT TECHNIQUE

TECHNICAL REPORT

CEI
IEC
62056-51

Première édition
First edition
1998-11

Comptage de l'électricité – Echange de données pour la lecture des compteurs, le contrôle des tarifs et de la charge –

Partie 51:
Protocoles de couche application

Electricity metering – Data exchange for meter reading, tariff and load control –

Part 51:
Application layer protocols

Numéro de référence
Reference number
CEI/IEC 62056-51:1998

Numéros des publications

Depuis le 1er janvier 1997, les publications de la CEI sont numérotées à partir de 60000.

Publications consolidées

Les versions consolidées de certaines publications de la CEI incorporant les amendements sont disponibles. Par exemple, les numéros d'édition 1.0, 1.1 et 1.2 indiquent respectivement la publication de base, la publication de base incorporant l'amendement 1, et la publication de base incorporant les amendements 1 et 2.

Validité de la présente publication

Le contenu technique des publications de la CEI est constamment revu par la CEI afin qu'il reflète l'état actuel de la technique.

Des renseignements relatifs à la date de reconfirmation de la publication sont disponibles dans le Catalogue de la CEI.

Les renseignements relatifs à des questions à l'étude et des travaux en cours entrepris par le comité technique qui a établi cette publication, ainsi que la liste des publications établies, se trouvent dans les documents ci-dessous:

- **«Site web» de la CEI***
- **Catalogue des publications de la CEI**
  Publié annuellement et mis à jour régulièrement (Catalogue en ligne)*
- **Bulletin de la CEI**
  Disponible à la fois au «site web» de la CEI* et comme périodique imprimé

Terminologie, symboles graphiques et littéraux

En ce qui concerne la terminologie générale, le lecteur se reportera à la CEI 60050: *Vocabulaire Electrotechnique International* (VEI).

Pour les symboles graphiques, les symboles littéraux et les signes d'usage général approuvés par la CEI, le lecteur consultera la CEI 60027: *Symboles littéraux à utiliser en électrotechnique,* la CEI 60417: S*ymboles graphiques utilisables sur le matériel. Index, relevé et compilation des feuilles individuelles,* et la CEI 60617: *Symboles graphiques pour schémas.*

* Voir adresse «site web» sur la page de titre.

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.

RAPPORT TECHNIQUE – TYPE 2

TECHNICAL REPORT – TYPE 2

CEI
IEC
62056-51

Première édition
First edition
1998-11

Comptage de l'électricité – Echange de données pour la lecture des compteurs, le contrôle des tarifs et de la charge –

Partie 51:
Protocoles de couche application

Electricity metering – Data exchange for meter reading, tariff and load control –

Part 51:
Application layer protocols

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

CODE PRIX
PRICE CODE V

Pour prix, voir catalogue en vigueur
For price, see current catalogue

SOMMAIRE

CONTENTS

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

COMPTAGE DE L'ÉLECTRICITÉ – ÉCHANGE DE DONNÉES POUR LA LECTURE DES COMPTEURS, LE CONTRÔLE DES TARIFS ET DE LA CHARGE –

Partie 51: Protocoles de couche application

AVANT-PROPOS

1) La CEI (Commission Electrotechnique Internationale) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI, entre autres activités, publie des Normes internationales. Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible un accord international sur les sujets étudiés, étant donné que les Comités nationaux intéressés sont représentés dans chaque comité d'études.
3) Les documents produits se présentent sous la forme de recommandations internationales. Ils sont publiés comme normes, rapports techniques ou guides et agréés comme tels par les Comités nationaux.
4) Dans le but d'encourager l'unification internationale, les Comités nationaux de la CEI s'engagent à appliquer de façon transparente, dans toute la mesure possible, les Normes internationales de la CEI dans leurs normes nationales et régionales. Toute divergence entre la norme de la CEI et la norme nationale ou régionale correspondante doit être indiquée en termes clairs dans cette dernière.
5) La CEI n'a fixé aucune procédure concernant le marquage comme indication d'approbation et sa responsabilité n'est pas engagée quand un matériel est déclaré conforme à l'une de ses normes.
6) L'attention est attirée sur le fait que certains des éléments de la présente Norme internationale peuvent faire l'objet de droits de propriété intellectuelle ou de droits analogues. La CEI ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de propriété et de ne pas avoir signalé leur existence.

La tâche principale des comités d'études de la CEI est d'élaborer des Normes internationales. Exceptionnellement, un comité d'études peut proposer la publication d'un rapport technique de l'un des types suivants:

- type 1, lorsque, en dépit de maints efforts, l'accord requis ne peut être réalisé en faveur de la publication d'une Norme internationale;
- type 2, lorsque le sujet en question est encore en cours de développement technique ou lorsque, pour une raison quelconque, la possibilité d'un accord pour la publication d'une Norme internationale peut être envisagée pour l'avenir mais pas dans l'immédiat;
- type 3, lorsqu'un comité d'études a réuni des données de nature différente de celles qui sont normalement publiées comme Normes internationales, cela pouvant comprendre, par exemple, des informations sur l'état de la technique.

Les rapports techniques de types 1 et 2 font l'objet d'un nouvel examen trois ans au plus tard après leur publication afin de décider éventuellement de leur transformation en Normes internationales. Les rapports techniques de type 3 ne doivent pas nécessairement être révisés avant que les données qu'ils contiennent ne soient plus jugées valables ou utiles.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

ELECTRICITY METERING – DATA EXCHANGE FOR METER READING, TARIFF AND LOAD CONTROL –

Part 51: Application layer protocols

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 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 International Standard 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 report of one of the following types:

- type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;
- type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;
- type 3, when a technical committee has collected data of a different kind from that which is normally published as an International Standard, for example “state of the art”.

Technical reports of types 1 and 2 are subject to review within three years of publication to decide whether they can be transformed into International Standards. Technical reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

La CEI 62056-51, rapport technique de type 2, a été établie par le comité d'études 13 de la CEI: Equipements de mesure de l'énergie électrique et de commande des charges.

Le texte de ce rapport technique est issu des documents suivants:

Projet de comité	Rapport de vote
13/1131/CDV	13/1167/RVC

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de ce rapport technique.

Le présent document est publié dans la série des rapports techniques de type 2 (conformément au paragraphe G.3.2.2 de la partie 1 des Directives ISO/CEI) comme «norme prospective d'application provisoire» dans le domaine de l'échange de données pour la lecture des compteurs, le contrôle des tarifs et de la charge car il est urgent d'avoir des indications sur la meilleure façon d'utiliser les normes dans ce domaine afin de répondre à un besoin déterminé.

Ce document ne doit pas être considéré comme une «Norme internationale». Il est proposé pour une mise en œuvre provisoire, dans le but de recueillir des informations et d'acquérir de l'expérience quant à son application pratique. Il est de règle d'envoyer les observations éventuelles relatives au contenu de ce document au Bureau Central de la CEI.

Il sera procédé à un nouvel examen de ce rapport technique de type 2 trois ans au plus tard après sa publication, avec la faculté d'en prolonger la validité pendant trois autres années, de le transformer en Norme internationale ou de l'annuler.

Les annexes A, B, C, D et E font partie intégrante de ce rapport technique.

IEC 62056-51, which is a technical report of type 2, has been prepared by IEC technical committee 13: Equipment for electrical energy measurement and load control.

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

Committee draft	Report on voting
13/1131/CDV	13/1167/RVC

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

This document is issued in the type 2 technical report series of publications (according to G.3.2.2 of part 1 of the IEC/ISO Directives) as a “prospective standard for provisional application” in the field of data exchange for meter reading, tariff and load control, because there is an urgent requirement for guidance on how standards in this field should be used to meet an identified need.

This document is not to be regarded as an “International Standard”. It is proposed for provisional application so that information and experience of its use in practice may be gathered. Comments on the content of this document should be sent to IEC Central Office.

A review of this type 2 technical report will be carried out not later than three years after its publication, with the options of either extension for a further three years or conversion to an International Standard or withdrawal.

Annexes A, B, C, D and E form an integral part of this technical report.

COMPTAGE DE L'ÉLECTRICITÉ – ÉCHANGE DE DONNÉES POUR LA LECTURE DES COMPTEURS, LE CONTRÔLE DES TARIFS ET DE LA CHARGE –

Partie 51: Protocoles de couche application

1 Généralités

1.1 Domaine d'application

Le présent rapport technique décrit une architecture de couche application utilisée pour communiquer avec les équipements de comptage en général, quels que soient le support physique et les protocoles de couches basses qui y sont associés dans un modèle réduit à trois couches.

Le présent rapport technique spécifie les protocoles à mettre en oeuvre pour la couche application à l'exception du modèle DLMS (Distribution Line Message Specification), qui est couvert par la CEI 61334-4-41.

1.2 Références normatives

Les documents normatifs suivants contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour le présent rapport technique. Au moment de la publication, les éditions indiquées étaient en vigueur. Tout document normatif est sujet à révision et les parties prenantes aux accords fondés sur le présent rapport technique sont invitées à rechercher la possibilités d'appliquer les éditions les plus récentes des documents normatifs indiqués ci-après. Les membres de la CEI et de l'ISO possèdent le registre des Normes internationales en vigueur.

CEI 61334-4-41:1996, *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*

ISO/CEI 8824:1990, *Technologies de l'information – Interconnexion de systèmes ouverts. Spécification de la notation de syntaxe abstraite numéro 1 (ASN.1) (Publiée actuellement en anglais seulement et édition retenue à titre provisoire)*

2 Présentation générale

2.1 Vocabulaire de base

Toute communication fait intervenir deux équipements représentés par les expressions système Appelant et système Appelé. L'Appelant est le système qui décide d'initialiser une communication avec un équipement distant dit Appelé; ces dénominations restent valables pendant toute la durée de la communication.

Une communication est décomposée en un certain nombre de transactions. Chaque transaction se traduit par une émission de l'Emetteur vers le Récepteur. Au gré de l'enchaînement des transactions, les systèmes Appelant et Appelé jouent tour à tour le rôle d'Emetteur et de Récepteur.

ELECTRICITY METERING – DATA EXCHANGE FOR METER READING, TARIFF AND LOAD CONTROL –

Part 51: Application layer protocols

1 General

1.1 Scope

This technical report describes an architectured application layer used for communication with metering equipments in general, whatever the associated physical medium and lower layer protocols in a collapsed three-layer model are.

This technical report specifies the protocols to be applied for the application layer except the DLMS (Distribution Line Message Specification) model, which is already covered by IEC 61334-4-41.

1.2 Normative references

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

IEC 61334-4-41:1996, *Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 41: Application protocols – Distribution line message specification (DLMS)*

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

2 General description

2.1 Basic vocabulary

All communications involve two sets of equipment represented by the terms Caller system and Called system. The Caller is the system that decides to initiate a communication with a remote system known as the Called party; these denominations remain valid throughout the duration of the communication.

A communication is broken down into a certain number of transactions. Each transaction is represented by a transmission from the Transmitter to the Receiver. During the sequence of transactions, the Caller and Called systems take turns to act as Transmitter and Receiver.

Les termes Client et Serveur ont le même sens que dans le modèle DLMS (voir CEI 61334-4-41). Le Serveur est le système qui se comporte comme un VDE (voir CEI 61334-4-41) pour toute soumission de requête de service particulière. Le Client est le système qui utilise le Serveur dans un but spécifique à l'aide d'une ou plusieurs soumissions de requête de service.

Le schéma basé sur un Client Appelant et un Serveur Appelé correspond certainement au cas de figure le plus fréquent. Mais on peut aussi imaginer une communication basée sur le couple Serveur Appelant et Client Appelé, en particulier pour signaler l'occurrence d'une alarme urgente.

2.2 Sous-couches et protocoles

Le modèle de couche Application décrit dans le présent rapport technique adopte un découpage en trois sous-couches: Transport, Application et DLMS. Chacune de ces sous-couches fait l'objet d'un protocole dont le nom est indiqué au tableau 1.

Tableau 1 – Sous-couches et protocoles

Sous-couches	Protocoles
DLMS	DLMS+
Application	Application+
Transport	Transport+

Les sous-couches Transport et Application forment un ensemble homogène appelé LLAC (Logical Link Access Control).

Le protocole DLMS+ de la sous-couche DLMS est décrit dans la CEI 61334-4-41.

2.3 Langage de spécification

Dans le présent rapport technique, le protocole de chaque sous-couche est décrit par des transitions d'état représentées sous forme de tableaux. La syntaxe utilisée pour la constitution de ces tableaux est définie par un langage de spécification présenté à l'annexe A.

En cas de divergence d'interprétation entre une partie du texte et un tableau de transitions d'état, c'est toujours le tableau qui fait référence.

3 Sous-couche Transport

3.1 Protocole Transport+

Le protocole Transport+ de la sous-couche Transport est conçu pour supporter le multiplexage des connexions de transport. Il est strictement identique pour l'Appelant et pour l'Appelé (comportement totalement symétrique).

3.2 Généralités

La sous-couche Transport est la première à prendre en charge des connexions directes entre les systèmes aux extrémités des liaisons. On peut parler de liaison de bout en bout pour toutes les connexions établies à ce niveau ainsi que pour celles situées au-dessus. Cette notion de bout en bout indique que les entités de transport offrent des services complètement indépendants des réseaux physiques.

The terms Client and Server have the same meanings as in the DLMS model (see IEC 61334-4-41). The Server is the system that acts as a VDE (see IEC 61334-4-41) for the submission of all special service requests. The Client is the system that uses the Server for a specific purpose by means of one or more service requests.

The situation involving a Caller Client and a Called Server is undoubtedly the most frequent case, but a communication based on a Caller Server and a Called Client is also possible, in particular to report the occurrence of an urgent alarm.

2.2 Sub-layers and protocols

The Application layer model described in this technical report uses a breakdown into three sub-layers: Transport, Application and DLMS. Each of these sub-layers is the subject of a protocol whose name is given in the table 1.

Table 1 – Sub-layers and protocols

Sub-layers	Protocols
DLMS	DLMS+
Application	Application+
Transport	Transport+

The Transport and Application sub-layers set up a homogeneous package called LLAC (Logical Link Access Control).

The DLMS+ protocol of the DLMS sub-layer is described in IEC 61334-4-41.

2.3 Specification language

In this technical report, the protocol of each sub-layer is described by state transitions represented in the form of tables. The syntax used in making up these tables is defined by a specification language described in annex A.

In the event of a difference in interpretation between part of the text and a state transition table, the table is always taken as the reference.

3 Transport sub-layer

3.1 Transport+ protocol

The Transport+ protocol of the Transport sub-layer is designed to support the multiplexing of transport connections. It is strictly identical for the Caller and for the Called party (completely symmetrical behaviour).

3.2 General information

The Transport sub-layer is the first one to handle direct connections between the systems at the ends of the links. All the connections set up at this level and those at higher levels can be considered as end-to-end links. This end-to-end notion indicates that the transport entities offer services which are completely independent of the physical networks.

Les propriétés les plus importantes de la sous-couche Transport sont le transport de bout en bout (déjà mentionné), la transparence (toute configuration binaire doit être acceptée par le protocole de transport pour être délivrée sans modification, quel que soit son format ou sa taille) et la sélection d'une qualité de service. La notion de qualité de service n'est mentionnée ici que pour mémoire, dans la mesure où le protocole Transport+ est orienté sans connexion.

La sous-couche Transport est chargée de prendre les messages provenant de la sous-couche Application. Etant donné que ces messages ont une taille dictée par l'application, la sous-couche Transport doit les segmenter en paquets (appelés TPDU: unités de données du protocole de transport) et les transmettre à la sous-couche Transport en correspondance. Réciproquement, elle doit recevoir les paquets provenant de la sous-couche Transport en correspondance puis assembler ceux-ci afin de les restituer sous forme de messages cohérents à la sous-couche Application.

Le protocole Transport+ doit pouvoir transmettre en parallèle des données dans les deux sens Appelant-Appelé et Appelé-Appelant. En outre, la solution du multiplexage des connexions de transport sur le même circuit virtuel assure que plusieurs associations d'application peuvent coexister dans une même communication.

Quelle que soit leur origine, les TPDU sont émises en utilisant les services de la couche Liaison. Bien entendu, cette dernière ignore tout du multiplexage mis en oeuvre au niveau supérieur.

3.3 Classes de protocole de transport

L'ISO propose différents types de services de réseaux en fonction des erreurs résiduelles dans les transmissions aux niveaux inférieurs. Deux sortes d'erreurs sont identifiées: les erreurs signalées (par exemple déconnexion autoritaire avec indication d'erreur) et les erreurs non signalées (erreurs de transmission non détectées et non corrigées). Sur cette base, trois types de réseaux existent et sont indiqués au tableau 2.

Tableau 2 – Types de services de réseaux en fonction des erreurs résiduelles

Type	Définition
A	Taux acceptable d'erreurs résiduelles (signalées et non signalées)
B	Taux acceptable d'erreurs non signalées Taux inacceptable d'erreurs signalées
C	Taux inacceptable d'erreurs des deux types

Le protocole Transport+ s'inspire de la classe 2 de l'ISO qui suppose un réseau de type A, et qui est caractérisée par des fonctions de segmentation et d'assemblage avec multiplexage, mais sans reprise sur erreur ni contrôle de flux.

3.4 Services et primitives de service de transport

L'utilisateur du protocole Transport+ dispose des services et primitives de service donnés au tableau 3.

Tableau 3 – Services et primitives de service de transport

Service	Primitive
T_DATA	T_DATA.req(STSAP, DTSAP, Pr, TSDU) T_DATA.ind(STSAP, DTSAP, TSDU)
T_ABORT	T_ABORT.req(Strong) T_ABORT.ind(ErrorNb)

The most important properties of the Transport sub-layer are end-to-end transport (mentioned above), transparency (any binary configuration must be accepted by the transport protocol and delivered without modification, whatever its format or size) and selection of a quality of service. The notion of quality of service is mentioned here for information only, as the Transport+ protocol is oriented without connection.

The Transport sub-layer accepts the messages from the Application sub-layer. As the size of these messages is dictated by the application, the Transport sub-layer must segment them into packets (called TPDUs: transport protocol data units) and transmit them to the correspondent Transport sub-layer. Reciprocally, it must receive the packets from the correspondent Transport sub-layer and assemble them into coherent messages for the Application sub-layer.

The Transport+ protocol must be able to transmit data in parallel in both directions, Caller-Called and Called-Caller. Moreover, the multiplexing of transport connections on the same virtual circuit means that several application associations can coexist in a given communication.

Whatever their origin, the TPDUs are transmitted using the services of the Data Link layer. Of course, this sub-layer is not aware of the multiplexing implemented at the higher level.

3.3 Transport protocol classes

The ISO proposes different types of network services depending on the residual errors in the transmissions at the lower levels. Two sorts of error are identified: the reported errors (for example forced disconnection with error indication) and the unreported errors (undetected and uncorrected transmission errors). On this basis, there are three types of network, summarized in table 2.

Table 2 – Types of network services depending on residual errors

Type	Definition
A	Acceptable residual error rate (reported and unreported)
B	Acceptable rate of unreported errors Unacceptable rate of reported errors
C	Unacceptable rate of both types of errors

The Transport+ protocol is derived from ISO class 2 which supposes a type A network, and which is characterized by segmenting and assembly functions with multiplexing, but without resumption after error or flow control.

3.4 Transport services and service primitives

Table 3 services and service primitives are available to users of the Transport+ protocol.

Table 3 – Transport services and service primitives

Service	Primitive
T_DATA	T_DATA.req(STSAP, DTSAP, Pr, TSDU) T_DATA.ind(STSAP, DTSAP, TSDU)
T_ABORT	T_ABORT.req(Strong) T_ABORT.ind(ErrorNb)

Le rôle attribué à chaque primitive est le suivant:

– T_DATA.req(STSAP, DTSAP, Pr, TSDU) permet à la sous-couche Application de demander à la sous-couche Transport le transfert avec la priorité Pr [1)] d'un message TSDU depuis une adresse de transport source STSAP vers une adresse de transport destination DTSAP ou bien depuis une adresse de transport destination DTSAP vers une adresse de transport source STSAP;

– T_DATA.ind(STSAP, DTSAP, TSDU) permet à la sous-couche Transport d'informer la sous-couche Application de l'arrivée d'un message TSDU depuis une adresse de transport source STSAP vers une adresse de transport destination DTSAP ou bien depuis une adresse de transport destination DTSAP vers une adresse de transport source STSAP;

– T_ABORT.req(Strong) permet à la sous-couche Application de demander à la sous-couche Transport de mettre fin à son activité avec la priorité Strong [2)];

– T_ABORT.ind(ErrorNb) permet à la sous-couche Transport d'informer la sous-couche Application de l'occurrence d'une erreur fatale repérée par le numéro ErrorNb.

3.5 Description des unités de données du protocole transport (TPDU)

Les messages échangés entre entités de transport sont acheminés par segments dans des TPDU qui contiennent chacune un nombre entier d'octets.

Dans le protocole Transport+, il n'y a qu'un seul type de TPDU défini par les cinq champs suivants:

– TPDUType (DT+) : 3 bits

– End : 1 bit

– STSAP : 2 bits

– DTSAP : 10 bits

– Packet : 0 octet à MaxPktSize octets

3 bits	1 bit	2 bits	10 bits	0 octet à MaxPktSize octets
DT+	End	STSAP	DTSAP	Packet

Figure 1 – Structure d'un TPDU

Le champ DT+ est toujours codé "101"B.

Lorsqu'il est positionné à 1, le bit du champ End indique que l'unité de données correspondante est la dernière d'un message segmenté.

Le champ STSAP référence l'adresse de transport source et le champ DTSAP correspond à l'adresse de transport destination. Par convention, la valeur "0000000000"B est réservée au DTSAP du Serveur DLMS d'administration des protocoles de communication.

Enfin, le champ Packet correspond au segment de données courant.

_____

1) Le niveau de priorité Pr permet de différencier le traitement des services urgents tels que InformationReport (niveau Pr=1) de celui des autres services DLMS (niveau Pr=0).

2) Le paramètre Strong permet de différencier le traitement des erreurs fatales (Strong=1) de celui des autres demandes de déconnexion physique (Strong=0) initialisées par la sous-couche Application.

The role assigned to each primitive is as follows:

– T_DATA.req(STSAP, DTSAP, Pr, TSDU) enables the Application sub-layer to request the Transport sub-layer to transfer with the priority Pr [1)] a TSDU message from a source transport address STSAP to a destination transport address DTSAP or from a destination transport address DTSAP to a source transport address STSAP;

– T_DATA.ind(STSAP, DTSAP, TSDU) enables the Transport sub-layer to inform the Application sub-layer of the arrival of a TSDU message from a source transport address STSAP to a destination transport address DTSAP or from a destination transport address DTSAP to a source transport address STSAP;

– T_ABORT.req(Strong) enables the Application sub-layer to request the Transport sub-layer to terminate its activity with the priority Strong [2)];

– T_ABORT.ind(ErrorNb) enables the Transport sub-layer to inform the Application sub-layer of the occurrence of a fatal error identified by the number ErrorNb.

3.5 Description of transport protocol data units (TPDUs)

The messages exchanged between transport entities are routed by segments in TPDUs, each of which contains a whole number of octets.

In the Transport+ protocol, there is only one type of TPDU defined by the following five fields:

– TPDUType (DT+) : 3 bits

– End : 1 bit

– STSAP : 2 bits

– DTSAP : 10 bits

– Packet : 0 octet to MaxPktSize octets

3 bits	1 bit	2 bits	10 bits	0 octet to MaxPktSize octets
DT+	End	STSAP	DTSAP	Packet

Figure 1 – Format of a TPDU

The DT+ field is always encoded "101"B.

When it is set to 1, the bit of the End field indicates that the corresponding data unit is the last one of a segmented message.

The STSAP field contains the source transport address and the DTSAP field contains the destination transport address. By convention, the value "0000000000"B is reserved for the DTSAP of the communication protocols management DLMS Server.

Finally, the Packet field contains the current data segment.

_____

1) The priority level Pr differentiates the processing of emergency services such as InformationReport (level Pr=1) from that of the other DLMS services (level Pr=0).

2) The Strong parameter differentiates the processing of fatal errors (Strong=1) from that of the other physical disconnection requests (Strong=0) initialized by the Application sub-layer.

3.6 Paramètres de transport

En l'absence de phase de connexion explicite, le nombre de connexions de transport et la taille des tampons ne sont pas négociés. Les règles à respecter sont les suivantes:

– au maximum 4 096 connexions de transport repérées chacune par un couple (STSAP, DTSAP);

– un besoin d'espace mémoire pour l'ensemble des tampons en émission et en réception de toutes les connexions de transport actives ne dépassant pas les capacités de l'équipement distant.

La taille réelle de l'espace mémoire global dépend de chaque équipement mais ne doit pas être inférieure à 512 octets. Une variable DLMS d'administration contient la valeur correspondante. Cette variable, de nom BufferPoolSize, est accessible par l'intermédiaire du Serveur DLMS d'administration des protocoles de communication.

La valeur du nombre MaxPktSize de taille du champ Packet doit être ajustée à la capacité des trames de la couche Liaison.

3.7 Transitions d'état

La machine d'état de l'Appelant est strictement identique à celle de l'Appelé. Les deux systèmes sont tour à tour Emetteur et Récepteur de TSDU. A tout instant, il n'existe qu'une seule occurrence de cet automate dans chaque équipement.

Tableau 4 – Transitions d'état de Transport+

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Stopped	$true()	init()	Idle
Idle	T_DATA.req(STSAP, DTSAP, Pr, TSDU) & bufferpool(TSDU)	SMsg=TSDU	M.Sgt
Idle	DL_DATA.ind(Pr, TPDU) & check_sgt(TPDU) & bufferpool(TPDU) & not(last_sgt(TPDU))	tsap(TPDU, STSAP, DTSAP) RMsg[STSAP, DTSAP, Pr]= concat(RMsg[STSAP, DTSAP, Pr], extract_pkt(TPDU))	Idle
Idle	DL_DATA.ind(Pr, TPDU) & check_sgt(TPDU) & bufferpool(TPDU) & last_sgt(TPDU)	tsap(TPDU, STSAP, DTSAP) T_DATA.ind(STSAP, DTSAP, concat(RMsg[STSAP, DTSAP, Pr], extract_pkt(TPDU))) RMsg[STSAP, DTSAP, Pr]=""	Idle
Idle	(T_DATA.req(_, _, _, TSDU) & not(bufferpool(TSDU))) | (DL_DATA.ind(_, TPDU) & check_sgt(TPDU) & not(bufferpool(TPDU)))	T_ABORT.ind(ET-2F) DL_ABORT.req(Strong=1)	Stopped
Idle	DL_DATA.ind(_, TPDU) & not(check_sgt(TPDU))	T_ABORT.ind(ET-1F) DL_ABORT.req(Strong=1)	Stopped
Idle	T_ABORT.req(Strong)	DL_ABORT.req(Strong)	Stopped
Idle	DL_ABORT.ind(ErrorNb)	T_ABORT.ind(ErrorNb)	Stopped
M.Sgt	size(SMsg)>MaxPktSize	End=0 DL_DATA.req(Pr, concat(DT+, End, STSAP, DTSAP, substr(1, MaxPktSize, SMsg))) SMsg=substr(MaxPktSize+1, SMsg)	M.Sgt
M.Sgt	size(SMsg)<=MaxPktSize	End=1 DL_DATA.req(Pr, concat(DT+, End, STSAP, DTSAP, SMsg)) SMsg=""	Idle

3.6 Transport parameters

In the absence of an explicit connection stage, the number of transport connections and the size of the buffers are not negotiated. The following rules are observed:

– a maximum of 4 096 transport connections each identified by a (STSAP, DTSAP) pair;

– a memory space requirement for all the transmission and reception buffers of all the active transport connections that does not exceed the capacity of the remote equipment.

The actual size of the overall memory space depends on each equipment but shall not be less than 512 octets. A DLMS management variable, BufferPoolSize, contains the corresponding value. This variable is accessible through the communication protocols management DLMS Server.

The value of the Packet field size, MaxPktSize, shall be adjusted to the frame capacity of the Data Link layer.

3.7 State transitions

The state machine of the Caller is strictly identical to that of the Called system. The two systems are in turn Transmitter and Receiver of TSDUs. At any time, there is only one occurrence of this controller in each set of equipment.

Table 4 – Transport+ state transitions

Initial state	Triggering condition	Set of actions	Final state
Stopped	$true()	init()	Idle
Idle	T_DATA.req(STSAP, DTSAP, Pr, TSDU) & bufferpool(TSDU)	SMsg=TSDU	M.Sgt
Idle	DL_DATA.ind(Pr, TPDU) & check_sgt(TPDU) & bufferpool(TPDU) & not(last_sgt(TPDU))	tsap(TPDU, STSAP, DTSAP) RMsg[STSAP, DTSAP, Pr]= concat(RMsg[STSAP, DTSAP, Pr], extract_pkt(TPDU))	Idle
Idle	DL_DATA.ind(Pr, TPDU) & check_sgt(TPDU) & bufferpool(TPDU) & last_sgt(TPDU)	tsap(TPDU, STSAP, DTSAP) T_DATA.ind(STSAP, DTSAP, concat(RMsg[STSAP, DTSAP, Pr], extract_pkt(TPDU))) RMsg[STSAP, DTSAP, Pr]=""	Idle
Idle	(T_DATA.req(_, _, _, TSDU) & not(bufferpool(TSDU))) | (DL_DATA.ind(_, TPDU) & check_sgt(TPDU) & not(bufferpool(TPDU)))	T_ABORT.ind(ET-2F) DL_ABORT.req(Strong=1)	Stopped
Idle	DL_DATA.ind(_, TPDU) & not(check_sgt(TPDU))	T_ABORT.ind(ET-1F) DL_ABORT.req(Strong=1)	Stopped
Idle	T_ABORT.req(Strong)	DL_ABORT.req(Strong)	Stopped
Idle	DL_ABORT.ind(ErrorNb)	T_ABORT.ind(ErrorNb)	Stopped
M.Sgt	size(SMsg)>MaxPktSize	End=0 DL_DATA.req(Pr, concat(DT+, End, STSAP, DTSAP, substr(1, MaxPktSize, SMsg))) SMsg=substr(MaxPktSize+1, SMsg)	M.Sgt
M.Sgt	size(SMsg)<=MaxPktSize	End=1 DL_DATA.req(Pr, concat(DT+, End, STSAP, DTSAP, SMsg)) SMsg=""	Idle

Tableau 5 – Signification des états mentionnés au tableau 4

Etat	Signification
Stopped	Etat de démarrage commun à l'Appelant et à l'Appelé
Idle	Etat d'attente d'une requête de la sous-couche Application ou d'une indication de la couche Liaison
M.Sgt (Must Segment)	Etat dans lequel un message est en cours de segmentation sur une connexion de transport

Tableau 6 – Définition des procédures et des fonctions classées par ordre alphabétique

Procédure ou fonction	Définition
bufferpool(TPDU) ou bufferpool(TSDU)	Vérification que l'unité de données TPDU ou TSDU peut être stockée dans l'espace mémoire global défini par la variable d'administration BufferPoolSize
check_sgt(TPDU)	Vérification que le type de la TPDU reçue est égal à DT+
concat(MsgPart, Packet) ou concat(DT+, End, STSAP, DTSAP, Packet)	Construction d'une chaîne binaire par concaténation de plusieurs paramètres
extract_pkt(TPDU)	Extraction du champ Packet de la TPDU reçue
init()	Initialisation de DT+ à "101"B, de MaxPktSize et de chacun des éléments du tableau RMsg[] à "". Le tableau RMsg[] possède trois dimensions: la première repère le numéro de STSAP, la seconde repère le numéro de DTSAP et la troisième la priorité (0 ou 1)
last_sgt(TPDU)	Vérification que la TPDU reçue possède un champ End monté à 1
size(SMsg)	Calcul de la taille en octets du message SMsg
substr(Begin, Length, SMsg) ou substr(Begin, SMsg)	Extraction d'une partie du message SMsg à partir de l'octet Begin sur la longueur Length en octets. Si la longueur n'est pas précisée, la partie extraite correspond à la fin du message partiel
tsap(TPDU, STSAP, DTSAP)	Extraction des champs STSAP et DTSAP de la TPDU reçue

3.8 Répertoire et traitement des erreurs

Les erreurs sont répertoriées selon le codage suivant:

ET erreur de la sous-couche Transport

– séparateur

N numéro de l'erreur

F erreur fatale

Table 5 – Meaning of the states listed in table 4

State	Meaning
Stopped	Startup state common to the Caller and the Called party
Idle	State waiting for a request from the Application sub-layer or an indication from the Data Link layer
M.Sgt (Must Segment)	State in which a message is being segmented on a transport connection

Table 6 – Definition of the procedures and functions classified in alphabetical order

Procedure or function	Definition
bufferpool(TPDU) or bufferpool(TSDU)	Check that the data unit TPDU or TSDU can be stored in the overall memory space defined by the management variable BufferPoolSize
check_sgt(TPDU)	Check that the type of the TPDU received is DT+
concat(MsgPart, Packet) or concat(DT+, End, STSAP, DTSAP, Packet)	Construction of a binary string by concatenation of several parameters
extract_pkt(TPDU)	Extraction of the Packet field from the received TPDU
init()	Initialization of DT+ at "101"B, of MaxPktSize and of each element of the array RMsg[] at "". The array RMsg[] has three dimensions: the first indicates the STSAP number, the second indicates the DTSAP number and the third indicates the priority (0 or 1)
last_sgt(TPDU)	Check that the received TPDU has an End field raised to 1
size(SMsg)	Calculation of the size of the SMsg message in octets
substr(Begin, Length, SMsg) or substr(Begin, SMsg)	Extraction of part of the SMsg message from the Begin octet over the length Length in octets. If the length is not specified, the extracted part corresponds to the end of the partial message
tsap(TPDU, STSAP, DTSAP)	Extraction of the STSAP and DTSAP fields from the received TPDU

3.8 List and processing of errors

The errors are listed with the following codes:

ET error in the Transport sub-layer

– separator

N number of the error

F fatal error

Tableau 7 – Tableau récapitulatif des erreurs

ET-1F	TPDU incorrecte. Cette erreur ne peut avoir comme origine qu'un type de TPDU différent de DT+
	Cette erreur conduit à réinitialiser la sous-couche Transport après avoir informé la sous-couche Application et fait avorter la couche Liaison
ET-2F	Saturation de l'espace mémoire global (en réception ou en émission)
	Cette erreur conduit à réinitialiser la sous-couche Transport après avoir informé la sous-couche Application et fait avorter la couche Liaison

Toute occurrence de l'une de ces erreurs fatales est remontée localement grâce à la primitive de service T_ABORT.ind. La liste complète des numéros des erreurs fatales est fournie à l'annexe B.

4 Sous-couche Application

4.1 Protocole Application+

Le protocole Application+ de la sous-couche Application adopte un comportement asymétrique basé sur le modèle Client-Serveur de DLMS (voir CEI 61334-4-41).

4.2 Généralités

La sous-couche Application se trouve considérablement simplifiée par l'adoption de DLMS. Ce modèle permet, du point de vue de la communication, une abstraction de tout équipement réel en un ou plusieurs équipements virtuels appelés VDE (voir CEI 61334-4-41). Chaque VDE est composé d'objets virtuels classés par type et accessibles par l'intermédiaire de services spécifiques.

En intégrant le modèle DLMS dans une vision stratifiée des protocoles, le rôle principal de la sous-couche Application devient l'encapsulation et le transport dans les deux sens des PDU DLMS (voir CEI 61334-4-41). Cependant, il est également nécessaire d'assurer

– la connexion et la déconnexion physique, éventuellement,
– la gestion des associations d'application,
– l'adoption d'une syntaxe de transfert pour le codage des données,
– la sécurité des échanges.

4.3 Sécurité des échanges

La notion de sécurité des échanges comprend

– l'authentification du Client et du Serveur,
– la confidentialité des données échangées,
– le contrôle d'accès aux objets variables du Serveur.

L'authentification permet au Serveur de contrôler l'identité du Client afin de lui associer les bons droits d'accès. Lorsque cette authentification est mutuelle, le Client peut aussi contrôler le Serveur et détecter une possible substitution d'équipement.

La confidentialité des données est destinée à protéger les données échangées contre d'éventuelles tentatives de lecture illicite par un tiers.

Quant au contrôle d'accès aux variables DLMS du Serveur, c'est le VDE-Handler (voir CEI 61334-4-41) qui l'assure grâce à un système de protection vérifiant les tentatives de lecture ou d'écriture du Client.

Table 7 – Error summary table

ET-1F	TPDU incorrect. The only possible cause of this error is a TPDU type different from DT+
	This error results in reinitialization of the Transport sub-layer after informing the Application sub-layer and causing the Data Link layer to abort
ET-2F	Saturation of the overall memory space (reception or transmission)
	This error results in reinitialization of the Transport sub-layer after informing the Application sub-layer and causing the Data Link layer to abort

Any occurrence of one of these fatal errors is sent up locally by means of the service primitive T_ABORT.ind. The complete list of fatal error numbers is given in annex B.

4 Application sub-layer

4.1 Application+ protocol

The Application+ protocol of the Application sub-layer has an asymmetrical behaviour based on the DLMS Client-Server model (see IEC 61334-4-41).

4.2 General information

The Application sub-layer is considerably simplified by the adoption of DLMS. From the point of view of the communication, this model enables the abstraction of any real equipment as one or more virtual equipment sets called VDE (see IEC 61334-4-41). Each VDE comprises virtual objects classified by type and accessible by means of specific services.

By integrating the DLMS model into a stratified view of the protocols, the main role of the Application sub-layer becomes the encapsulation and the transport in both directions of DLMS PDUs (see IEC 61334-4-41). However, it is also necessary to ensure

– physical connection and disconnection, if applicable,
– management of application associations,
– selection of a transfer syntax for data encoding,
– security of exchanges.

4.3 Security of exchanges

The security of exchanges includes

– authentication of the Client and the Server,
– confidentiality of exchanged data,
– control of access to the variable objects of the Server.

The authentication enables the Server to control the identity of the Client in order to provide him with the proper access rights. When this authentication is mutual, the Client can also control the Server and detect a possible substitution of equipment.

The confidentiality of data is designed to protect the exchanged data against possible unauthorized readings.

The VDE-Handler (see IEC 61334-4-41) provides access to the DLMS variables of the Server by means of a protection system controlling the Client read or write attempts.

4.4 Authentification du Client et du Serveur

L'authentification mutuelle est assurée par un double échange de messages cryptés durant la phase de demande d'activation d'association d'application en provenance du Client.

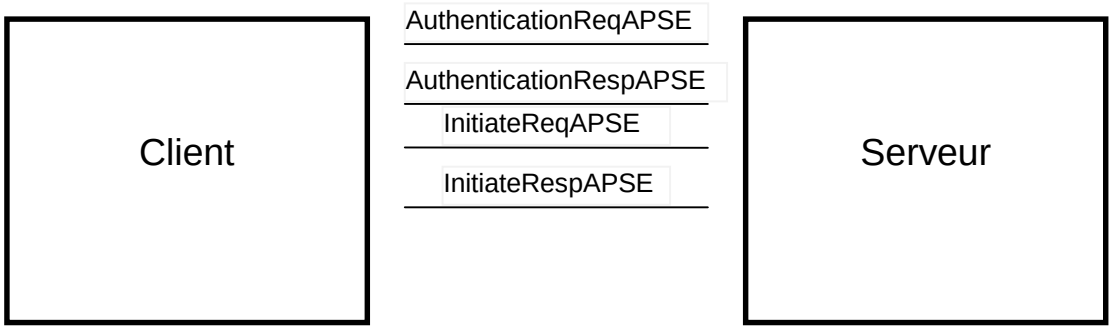

Figure 2 – Echanges d'authentification

Chaque équipement doit disposer d'un système de génération de nombres aléatoires.

L'authentification est obtenue par un échange de nombres aléatoires cryptés grâce à une clef secrète spécifique à chaque type de Client (Client-type). Les nombres aléatoires sont définis sur 8 octets (voir annexe C) et ils sont cryptés grâce à un algorithme utilisant une clef de cryptage Ki sur 8 octets connue à la fois du Client et du Serveur.

Sur une demande d'Initiate, un nombre aléatoire Nc est d'abord généré par le Client et transmis dans l'AuthenticationReqAPSE.

A l'arrivée sur le Serveur, Nc est crypté avec le même algorithme et la clef Ki pour obtenir le nombre aléatoire crypté NcK. Débute alors la séquence interne d'authentification composée d'un échange de deux PDU.

Le premier PDU (AuthenticationRespAPSE envoyé par le Serveur au Client) contient ce nombre aléatoire crypté NcK et un nombre aléatoire Ns généré par le Serveur.

A la réception de ce PDU, le Client compare NcK à un nombre Nc' obtenu en cryptant le nombre aléatoire Nc en utilisant le même algorithme avec la clef Ki. Si Nc' = NcK, le Client considère alors que le Serveur appelé est authentifié. Sinon, il suppose qu'un imposteur essaie de communiquer sur la ligne à la place de l'unité désirée et il ferme la communication.

Après avoir authentifié le Serveur, le Client crypte le nombre aléatoire Ns avec le même algorithme et la clef Ki pour obtenir le nombre aléatoire crypté NsK et le transmettre ensuite dans le second PDU, InitiateReqAPSE.

A la réception de ce PDU, le Serveur compare NsK à un nombre Ns' obtenu en cryptant le nombre aléatoire Ns en utilisant le même algorithme avec la clef Ki. Si Ns' = NsK, alors le Serveur considère que le Client est authentifié. Sinon, il suppose qu'un imposteur essaie de communiquer sur la ligne et ferme la communication.

Sur chaque équipement Client et sur chaque équipement Serveur homologue, il existe une table associant une clef privée à chaque valeur possible de Client-type. Il n'y a donc aucun échange de clef pendant la communication.

4.4 Authentication of Client and Server

Mutual authentication is provided by a double exchange of ciphered messages during the phase of application association activation request from the Client.

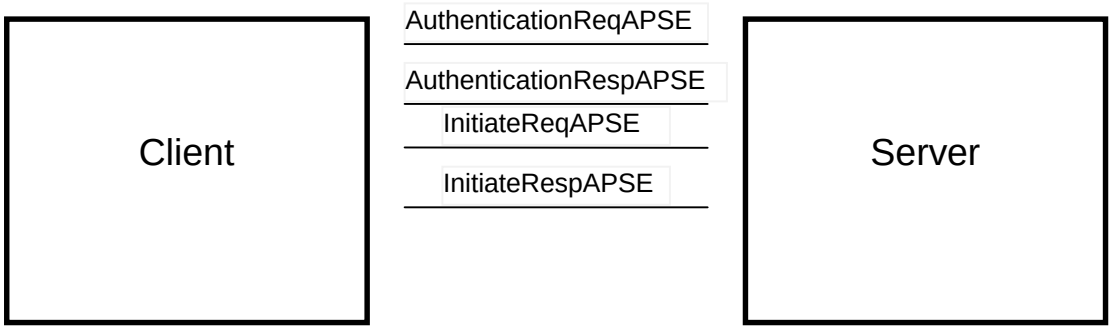

Figure 2 – Authentication exchanges

Each equipment shall have a random number generation system.

Authentication is carried out by an exchange of random numbers ciphered thanks to a secret key specific to each type of Client (Client-type). The random numbers are defined in 8 octets (see annex C) and they are ciphered thanks to an algorithm using an 8-octet ciphering key Ki known both to the Client and the Server.

On an Initiate request, a random number Nc is first generated by the Client and transmitted into the AuthenticationReqAPSE.

On arrival at the Server, Nc is ciphered by the same algorithm with key Ki to get the ciphered random number NcK. Then the internal sequence for authentication occurs, which consists in an exchange of two PDUs.

The first PDU (AuthenticationRespAPSE from Server to Client) contains this ciphered random number NcK and a random number Ns generated by the Server.

On reception of this PDU, the Client compares NcK to an Nc' number obtained by ciphering the random number Nc using the same algorithm with key Ki. If Nc’ = NcK; the Client then considers the called Server as authenticated. Otherwise, it considers that an impostor is attempting to speak on the line in place of the wanted unit and it aborts the communication.

After correct authentication of the Server, the Client ciphers the random number Ns by the same algorithm with key Ki to get the ciphered random number NsK and then transmit it into the second PDU, InitiateReqAPSE.

On reception of this PDU, the Server compares NsK to an Ns' number obtained by ciphering the random number Ns using the same algorithm with key Ki. If Ns’ = NsK, then the Server considers the Client as authenticated. Otherwise, it considers that an impostor is attempting to speak on the line and it aborts the communication.

On all Client and opposite Server equipments, there is a table joining a private key to each possible value of Client-type. Therefore, there is no exchange of key during the communication.

4.5 Confidentialité des données échangées

La confidentialité des données échangées est assurée en brouillant chaque message à protéger. L'algorithme de brouillage et de débrouillage utilisé par le protocole Application+ pour les échanges autres que l'authentification est décrit en détail à l'annexe D. Ses caractéristiques principales sont les suivantes:

– le code est simple à élaborer et plus rapide à l'exécution que le code d'authentification;
– la fonction de brouillage est pseudo-aléatoire et de période longue;
– la fonction de débrouillage est la même que la fonction de brouillage (opération involutive);
– le seul paramètre de l'algorithme est une clef de brouillage de n bits calculée automatiquement par le Serveur pour chaque association d'application.

4.6 Contexte d'application

Le protocole Application+ comprend un seul SASE (Specific Application Service Element) dont le nom est APSE. Le APSE se divise en Client APSE et Server APSE.

En outre, pour qu'un échange puisse s'effectuer correctement, il faut que le Client et le Serveur travaillent avec le même contexte d'application (ensemble de règles régissant l'échange d'information sur une association d'application). Le protocole Application+ permet la négociation de ce contexte d'application. Il existe, néanmoins, un contexte d'application par défaut (identifié par la valeur entière 0) connu a priori du Client et du Serveur.

Une variable DLMS d'administration contient la liste des contextes d'application supportés par le Serveur. Cette variable, de nom ApplicationContextNameList, est accessible par l'intermédiaire du Serveur DLMS d'administration des protocoles de communication.

La liste des règles implicites du contexte d'application par défaut du APSE est la suivante:

– choix d'un profil de communication spécifique pour les couches basses;
– spécification de la syntaxe abstraite utilisée (voir ISO 8824);
– spécification des règles de codage et de décodage;
– liste des ASE impliqués (APSE) et description des APDU;
– mécanisme d'authentification par cryptage, ainsi que méthode de calcul des nombres aléatoires définie à l'annexe C;
– algorithme de brouillage des échanges défini à l'annexe D, ainsi que mode de calcul de la clef de brouillage;
– mode de brouillage spécifique à chaque classe de service DLMS en fonction du type d'identifiant client (voir annexe E).

On notera que le contexte d'application par défaut ne contient aucun algorithme de compression de données.

4.5 Confidentiality of exchanged data

The confidentiality of exchanged data is possible by masking each message to be protected. The masking and unmasking algorithm used by the Application+ protocol for exchanges other than authentication is described thoroughly in annex D. Its main characteristics are the following:

– the code is simple to develop and its execution is quicker than the authentication code;
– the masking function is pseudo-random and of long period;
– the masking and unmasking functions are identical (involutive operation);
– the only parameter of the algorithm is a n bit masking key automatically calculated by the Server for each application association.

4.6 Application context

The Application+ protocol includes a single SASE (Specific Application Service Element) called APSE. APSE is divided into Client APSE and Server APSE.

In addition, for an exchange to be executed correctly, the Client and the Server must work with the same application context (set of rules governing the exchange of information on an application association). The Application+ protocol enables the negotiation of this application context. However, there is a default application context (identified by the whole number 0) known a priori by the Client and the Server.

A management DLMS variable, ApplicationContextNameList, contains the list of the application contexts supported by the Server. This variable is accessible through the communication protocols management DLMS Server.

The implicit rules of the APSE default application context are the following:

– choice of a specific communication profile for the lower layers;
– specification of the abstract syntax used (see ISO 8824);
– specification of the encoding and decoding rules;
– list of the ASEs involved (APSE) and description of APDUs;
– ciphering authentication mechanism and random number generation procedure described in annex C;
– exchange masking algorithm described in annex D as well as masking key generation procedure;
– masking mode specific to each DLMS service class depending on the Client identifier type (see annex E).

It should be noted that the default application context does not contain a data compression algorithm.

4.7 Contexte DLMS

Les services DLMS ne transportent aucune information d'identification réciproque du Client et du Serveur. En fait, ces informations sont directement contenues dans le contexte DLMS. Un contexte DLMS comprend les éléments suivants:

– un contexte d'application tel que défini précédemment;
– un STSAP (TSAP Source);
– un DTSAP (TSAP Destination);
– le type du Client Client-type (voir annexe E);
– côté Client Appelant, l'adresse physique du Serveur Appelé;
– côté Serveur Appelant, la liste des adresses physiques des Clients appelables ainsi que l'identifiant du Serveur Server-identifier (voir annexe E);
– si le Client est l'Appelant, l'adresse physique du Client;
– la valeur confidentielle de la clef d'authentification;
– la valeur confidentielle de la clef de brouillage des échanges.

En général, l'Appelant est un Client-type autorisé et l'Appelé un VDE Serveur caractérisé par son adresse transport DTSAP. Toutefois, le contraire se produit lorsque des alarmes urgentes doivent être remontées au Client. Dans ce cas, l'initialisation de l'association d'application est une initiative du Serveur.

C'est toujours le Client-type qui repère sans ambiguïté un contexte DLMS. Cette notion correspond très exactement à celle de VAAName (voir CEI 61334-4-41). Dans le protocole Application+, les VAA sont considérées comme des objets statiques.

Un seul contexte DLMS est actif à un instant donné pour chaque connexion de transport active. Côté Client Appelant, le contexte courant est supposé avoir déjà été activé par des moyens locaux à chaque équipement et non décrits ici. Côté Serveur Appelé, en revanche, le contexte est activé grâce au service Initiate (voir CEI 61334-4-41) et inactivé par une indication d'Abort (voir CEI 61334-4-41).

4.8 Services et primitives de service d'application

L'utilisateur du protocole Application+ dispose de la totalité des services et primitives de service définis dans la spécification DLMS, à l'exception des services non confirmés.

Côté Client, il est également nécessaire de disposer d'une requête de service non confirmé A_Disconnect.req() afin de pouvoir, éventuellement, effectuer la déconnexion physique.

Pour la définition complète des services, primitives de service et PDU DLMS, se reporter à la CEI 61334-4-41.

4.9 Description des unités de données du protocole application (APDU)

Les messages échangés sur une association d'application contiennent chacun un nombre entier d'octets qui, en tout état de cause, doit toujours rester compatible avec la taille de l'espace mémoire global de la sous-couche Transport.

4.7 DLMS context

The DLMS services do not transport any Client or Server reciprocal identification information. This information is, in fact, contained directly in the DLMS context. A DLMS context contains the following elements:

– an application context as defined above;
– an STSAP (Source TSAP);
– a DTSAP (Destination TSAP);
– the Client-type (see annex E);
– at the Caller Client end, the physical address of the Called Server;
– at the Caller Server end, the list of the physical addresses of the Clients that can be called and the Server identifier Server-identifier (see annex E);
– if the Client is the Caller, the physical address of the Client;
– the confidential value of the authentication key;
– the confidential value of the masking key.

In general, the Caller is an authorized Client-type and the Called system a VDE Server identified by its transport address DTSAP. However, the converse is the case when emergency alarms must be communicated to the Client. In this case, the application association is initialized by the Server.

The Client-type always identifies a DLMS context unambiguously. This notion corresponds exactly to that of VAAName (see IEC 61334-4-41). In the Application+ protocol, the VAAs are considered as static objects.

Only one DLMS context is active at a given time for each active transport connection. At the Caller Client end, the current context is assumed to have been already activated by means local to each equipment system and not described here. In contrast, at the Called Server end, the context is activated by means of the Initiate service (see IEC 61334-4-41) and inactivated by an Abort indication (see IEC 61334-4-41).

4.8 Application services and service primitives

All the services and service primitives defined in the DLMS specification are available to the users of the Application+ protocol, with the exception of unconfirmed services.

At the Client end, an unconfirmed service request A_Disconnect.req() is also required to be able to carry out a physical disconnection, if applicable.

For a complete definition of services, service primitives and DLMS PDUs, see IEC 61334-4-41.

4.9 Description of application protocol data units (APDUs)

Each of the messages exchanged on an application association contains a whole number of octets which, in all cases, shall always remain compatible with the size of the overall memory space of the Transport sub-layer.

Dans le protocole Application+, il y a dix types d'APDU comme l'indique la définition ASN.1 [3] suivante.

```
APSEPDU::=CHOICE {
    confirmedRequest          [0]   ConfirmedReqAPSE,
    confirmedResponse         [1]   ConfirmedRespAPSE,
    confirmedError            [2]   ConfirmedErrorAPSE,
    unsolicitedRequest        [3]   UnsolicitedReqAPSE,
    authenticationRequest     [4]   AuthenticationReqAPSE,
    authenticationResponse    [5]   AuthenticationRespAPSE,
    initiateRequest           [6]   InitiateReqAPSE,
    initiateResponse          [7]   InitiateRespAPSE,
    initiateError             [8]   InitiateErrorAPSE,
    abortRequest              [9]   AbortReqAPSE }

ConfirmedReqAPSE::=OCTET STRING
 -- obtenu après cryptage éventuel d'une PDU DLMS de type ConfirmedServiceRequest,
 -- GetStatusRequest, GetNameListRequest, GetVariableAttributeRequest, ReadRequest ou
 -- WriteRequest type

ConfirmedRespAPSE::=OCTET STRING
 -- obtenu après cryptage éventuel d'une PDU DLMS de type ConfirmedServiceResponse,
 -- GetStatusResponse, GetNameListResponse, GetVariableAttributeResponse,
ReadResponse ou
 -- WriteResponse type

ConfirmedErrorAPSE::=OCTET STRING
 -- obtenu directement à partir d'une PDU DLMS de type ConfirmedServiceError

UnsolicitedReqAPSE::=SEQUENCE {
server-identifier                    OCTET STRING,
client-type                          INTEGER(-32 768..32 767),
unsolicited-service-request          OCTET STRING
 -- obtenu après cryptage éventuel d'une PDU DLMS de type UnsolicitedServiceRequest ou
 -- InformationReportRequest -- }

AuthenticationReqAPSE::=SEQUENCE {
client-type                          INTEGER(-32 768..32 767),
client-random-number                 BIT STRING(SIZE(64)) }

AuthenticationRespAPSE::=SEQUENCE {
ciphered-transformed-client-random-number    BIT STRING(SIZE(64)),
server-random-number                 BIT STRING(SIZE(64)) }

InitiateReqAPSE::=SEQUENCE {
ciphered-transformed-server-random-number    BIT STRING(SIZE(64)),
proposed-app-ctx-name                INTEGER(0..255),
calling-physical-address             OCTET STRING,
initiate-request                     OCTET STRING
 -- obtenu directement à partir d'une PDU DLMS de type InitiateRequest -- }

InitiateRespAPSE::=SEQUENCE {
negociated-app-ctx-name              INTEGER(0..255),
initiate-response                    OCTET STRING
 -- obtenu directement à partir d'une PDU DLMS de type InitiateResponse -- }
```

———————

[3] ASN: Notation de Syntaxe Abstraite.

In the Application+ protocol, there are ten types of APDU as indicated in the following ASN.1 [3] definition below.

```
APSEPDU::=CHOICE {
    confirmedRequest          [0]   ConfirmedReqAPSE,
    confirmedResponse         [1]   ConfirmedRespAPSE,
    confirmedError            [2]   ConfirmedErrorAPSE,
    unsolicitedRequest        [3]   UnsolicitedReqAPSE,
    authenticationRequest     [4]   AuthenticationReqAPSE,
    authenticationResponse    [5]   AuthenticationRespAPSE,
    initiateRequest           [6]   InitiateReqAPSE,
    initiateResponse          [7]   InitiateRespAPSE,
    initiateError             [8]   InitiateErrorAPSE,
    abortRequest              [9]   AbortReqAPSE }

ConfirmedReqAPSE::=OCTET STRING
 -- obtained after ciphering, if any, of a DLMS PDU of the ConfirmedServiceRequest,
 -- GetStatusRequest, GetNameListRequest, GetVariableAttributeRequest, ReadRequest or
 -- WriteRequest type

ConfirmedRespAPSE::=OCTET STRING
 -- obtained after ciphering, if any, of a DLMS PDU of the ConfirmedServiceResponse,
 -- GetStatusResponse, GetNameListResponse, GetVariableAttributeResponse,
ReadResponse or
 -- WriteResponse type

ConfirmedErrorAPSE::=OCTET STRING
 -- obtained directly from a DLMS PDU of the ConfirmedServiceError type

UnsolicitedReqAPSE::=SEQUENCE {
server-identifier                          OCTET STRING,
client-type                                INTEGER(-32 768..32 767),
unsolicited-service-request                OCTET STRING
 -- obtained after ciphering, if any, of a DLMS PDU of the UnsolicitedServiceRequest or
 -- InformationReportRequest type -- }

AuthenticationReqAPSE::=SEQUENCE {
client-type                                INTEGER(-32 768..32 767),
client-random-number                       BIT STRING(SIZE(64)) }

AuthenticationRespAPSE::=SEQUENCE {
ciphered-transformed-client-random-number  BIT STRING(SIZE(64)),
server-random-number                       BIT STRING(SIZE(64)) }

InitiateReqAPSE::=SEQUENCE {
ciphered-transformed-server-random-number  BIT STRING(SIZE(64)),
proposed-app-ctx-name                      INTEGER(0..255),
calling-physical-address                   OCTET STRING,
initiate-request                           OCTET STRING
 -- obtained directly from a DLMS PDU of the InitiateRequest type -- }

InitiateRespAPSE::=SEQUENCE {
negociated-app-ctx-name                    INTEGER(0..255),
initiate-response                          OCTET STRING
 -- obtained directly from a DLMS PDU of the InitiateResponse type -- }
```

———————

[3] ASN: Abstract Syntax Notation.

InitiateErrorAPSE::=OCTET STRING
-- obtenu directement à partir d'une PDU DLMS de type ConfirmedServiceError

AbortReqAPSE::=OCTET STRING
-- obtenu directement à partir d'une PDU DLMS de type AbortRequest

4.10 Gestion des échanges

Le protocole Application+ n'est pas symétrique puisque les rôles du Client et du Serveur ne sont pas interchangeables.

Lorsqu'aucune connexion de transport n'est encore active côté Client, l'initialisation d'une association d'application (requête de service Initiate de DLMS) prend en charge une éventuelle connexion physique.

Dans tous les cas, il y a un échange de messages d'authentification et un contrôle du type du Client par le Serveur. Si l'authentification a réussi, cette information associée à la valeur du DTSAP permet au Serveur de réactiver l'association d'application et le contexte DLMS idoines; elle permet aussi d'en déduire le VAA approprié.

Conformément à DLMS, le protocole Application+ n'admet pas de requête pendante. Après l'envoi d'une requête de service confirmé, la sous-couche Application du Client attend toujours une réponse venant de la sous-couche Application du Serveur avant d'émettre à nouveau. Cette attente est contrôlée par le réveil MVRT (Maximum VDE Response Time), qu'il convient de considérer comme un simple chien de garde logiciel armé pour une durée T2.

Après la phase d'authentification, toute erreur de cryptage détectée par le Serveur est assimilée à une tentative de violation et conduit donc à l'arrêt de la communication au niveau Application. En revanche, une erreur de cryptage n'est pas considérée comme fatale côté Client.

Le niveau de priorité Pr permet de différencier le traitement des services non sollicités qui sont considérés comme urgents (niveau Pr=1) de celui des autres services DLMS (niveau Pr=0).

Le paramètre Strong permet de différencier le traitement des erreurs fatales (Strong=1) de celui des autres demandes de déconnexion physique (Strong=0) initialisées par la sous-couche Application.

Enfin, côté Client, le service non confirmé A_Disconnect.req peut solliciter une déconnexion physique conduisant à une libération du support physique et à la réinitialisation de toutes les occurrences d'automate d'application.

4.11 Paramètre d'application

Le temps d'attente maximal, par le Client, d'une réponse en retour du Serveur avant déconnexion logique doit être choisi tel que

T2 > RespTime + MaxReqTime + MaxRespTime
où RespTime représente le temps de réponse théorique du Serveur, MaxReqTime le temps de transmission maximal d'une requête de service et MaxRespTime le temps de transmission maximal d'une réponse de service.

4.12 Transitions d'état

Les machines d'état du Client et du Serveur sont différentes. En outre, pour des raisons de clarté, chaque machine est découpée en trois tableaux: la gestion de contexte, la gestion des services confirmés et la gestion des services non sollicités. De plus, dans chaque équipement, il existe une occurrence de l'automate par connexion de transport active.

InitiateErrorAPSE::=OCTET STRING
-- obtained directly from a DLMS PDU of the ConfirmedServiceError type

AbortReqAPSE::=OCTET STRING
-- obtained directly from a DLMS PDU of the AbortRequest type

4.10 Management of exchanges

The Application+ protocol is not symmetrical since the roles of the Client and the Server are not interchangeable.

When no transport connection is active yet at the Client end, the initialization of an application association (DLMS Initiate service request) takes over a possible physical connection.

In all cases, there is an authentication message exchange and a Client type control by the Server. If the authentication has succeeded, this information, together with the DTSAP value, enables the Server to reactivate the proper application association and DLMS context as well as to deduce the appropriate VAA.

In accordance with DLMS, the Application+ protocol does not accept pending requests. After sending a confirmed service request, the Application sub-layer of the Client always waits for a response from the Application sub-layer of the Server before transmitting again. This wait is controlled by the wakeup MVRT (Maximum VDE Response Time), which should be considered as a simple software watchdog set for a time T2.

After the authentication phasis, any ciphering error detected by the Server is treated as a violation attempt and thus causes the communication to stop at the Application level. In contrast, a ciphering error is not considered as fatal at the Client end.

The priority level Pr differentiates the processing of unsolicited services which are considered as urgent (level Pr=1) from that of the other DLMS services (level Pr=0).

The Strong parameter differentiates the processing of fatal errors (Strong=1) from that of the other physical disconnection requests (Strong=0) initialized by the Application sub-layer.

Finally, at the Client end, the unconfirmed service A_Disconnect.req can solicit a physical disconnection leading to the freeing of the medium and reinitialization of all the occurrences of the application controller.

4.11 Application parameter

The maximum time waited by the Client for the return message from the Server before logical disconnection shall be chosen such that

> T2 > RespTime + MaxReqTime + MaxRespTime
> where RespTime represents the theoretical response time of the Server, MaxReqTime the maximum time for transmission of a service request and MaxRespTime the maximum time for transmission of a service response.

4.12 State transitions

The state machines of the Client and the Server are different. Moreover, for clarity, each machine is broken down into three tables: context management, confirmed service management and unsolicited service management. In addition, in each equipment system, there is one occurrence of the controller per active transport connection.

Tableau 8 – Tableau de gestion du contexte côté Serveur

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Locked	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, AuthenticationReqAPSE) & extract_authentication_req(APSEPDU, ClientType) & check_client_type(DTSAP, ClientType)	APSEPDU=build_tasepdu(Authentication RespAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	IR.E
Locked	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked
IR.E	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, InitiateReqAPSE)	Extract_OK=extract_initiate_req(APSEPDU, AppCtx, CallingAddr, DLMSPDU)	Extract
IR.E	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked
Extract	Extract_OK	set_dlms_context(AppCtx, STSAP, DTSAP, ClientType, CallingAddr) Initiate.ind(DLMSPDU)	Con.E
Extract	not(Extract_OK)	$none()	Locked
Con.E	Initiate.rsp(DLMSPDU) & dlmspdu_type(DLMSPDU, initiate) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(InitiateRespAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	Idle
Con.E	Initiate.rsp(DLMSPDU) & dlmspdu_type(DLMSPDU, initiate-error) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(InitiateErrorAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	Locked
Con.E	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked
Idle	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, AuthenticationReqAPSE) & extract_authentication_req(APSEPDU, ClientType) & check_client_type(DTSAP, ClientType)	build_evt(AbortIndication) APSEPDU=build_tasepdu(Authentication RespAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	IR.E
Idle	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, AbortReqAPSE)	Abort.ind()	Locked
Idle	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked

Table 8 – Server end context management table

Initial state	Triggering condition	Set of actions	Final state
Locked	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, AuthenticationReqAPSE) & extract_authentication_req(APSEPDU, ClientType) & check_client_type(DTSAP, ClientType)	APSEPDU=build_tasepdu(Authentication RespAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	IR.E
Locked	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked
IR.E	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, InitiateReqAPSE)	Extract_OK=extract_initiate_req(APSEPDU, AppCtx, CallingAddr, DLMSPDU)	Extract
IR.E	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked
Extract	Extract_OK	set_dlms_context(AppCtx, STSAP, DTSAP, ClientType, CallingAddr) Initiate.ind(DLMSPDU)	Con.E
Extract	not(Extract_OK)	$none()	Locked
Con.E	Initiate.rsp(DLMSPDU) & dlmspdu_type(DLMSPDU, initiate) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(InitiateRespAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	Idle
Con.E	Initiate.rsp(DLMSPDU) & dlmspdu_type(DLMSPDU, initiate-error) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(InitiateErrorAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	Locked
Con.E	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked
Idle	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, AuthenticationReqAPSE) & extract_authentication_req(APSEPDU, ClientType) & check_client_type(DTSAP, ClientType)	build_evt(AbortIndication) APSEPDU=build_tasepdu(Authentication RespAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	IR.E
Idle	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, AbortReqAPSE)	Abort.ind()	Locked
Idle	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked

Tableau 9 – Tableau de gestion du contexte côté Client

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Locked	Initiate.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & address(CallingAddr, CalledAddr)	Connect_OK=client_connect(CallingAddr, CalledAddr)	Connect
Locked	A_Disconnect.req()	T_ABORT.req(Strong=0) $purge()	Locked
Connect	Connect_OK	APSEPDU=build_tasepdu(Authentication ReqAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) init_timer(T2)	AC.T
Connect	not(Connect_OK)	DLMSPDU=service_error(application-unreachable) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Locked
AC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, AuthenticationRespAPSE)	D-decipher-OK= d-decipher(AuthenticationRespAPSE, DKey)	D.Deciph
AC.T	time_out()	DLMSPDU=service_error(time-elapsed) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Locked
AC.T	(T_ABORT.ind(ErrorNb) & last_association()) | (exist_abort_ind() & not(last_association()))	DLMSPDU=service_error(provider-communication-error) Initiate.cnf(DLMSPDU) store_error(ErrorNb) inactivate_dlms_context()	Locked
D.Deciph	D-decipher-OK	APSEPDU=build_tasepdu(InitiateReqAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) update_dlms_context(_, DKey)	NC.T
D.Deciph	not(D-decipher-OK)	stop_timer() DLMSPDU=service_error(deciphering-error) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Dis
NC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, InitiateRespAPSE)	stop_timer() Extract_OK=extract_initiate_resp(APSEPDU, AppCtx, DLMSPDU)	Extract
NC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, InitiateErrorAPSE) & convert(APSEPDU, DLMSPDU)	stop_timer() Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Dis
NC.T	time_out()	DLMSPDU=service_error(time-elapsed) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Locked

Table 9 – Client end context management table

Initial state	Triggering condition	Set of actions	Final state
Locked	Initiate.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & address(CallingAddr, CalledAddr)	Connect_OK=client_connect(CallingAddr, CalledAddr)	Connect
Locked	A_Disconnect.req()	T_ABORT.req(Strong=0) $purge()	Locked
Connect	Connect_OK	APSEPDU=build_tasepdu(Authentication ReqAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) init_timer(T2)	AC.T
Connect	not(Connect_OK)	DLMSPDU=service_error(application-unreachable) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Locked
AC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, AuthenticationRespAPSE)	D-decipher-OK= d-decipher(AuthenticationRespAPSE, DKey)	D.Deciph
AC.T	time_out()	DLMSPDU=service_error(time-elapsed) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Locked
AC.T	(T_ABORT.ind(ErrorNb) & last_association()) | (exist_abort_ind() & not(last_association()))	DLMSPDU=service_error(provider-communication-error) Initiate.cnf(DLMSPDU) store_error(ErrorNb) inactivate_dlms_context()	Locked
D.Deciph	D-decipher-OK	APSEPDU=build_tasepdu(InitiateReqAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) update_dlms_context(_, DKey)	NC.T
D.Deciph	not(D-decipher-OK)	stop_timer() DLMSPDU=service_error(deciphering-error) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Dis
NC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, InitiateRespAPSE)	stop_timer() Extract_OK=extract_initiate_resp(APSEPDU, AppCtx, DLMSPDU)	Extract
NC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, InitiateErrorAPSE) & convert(APSEPDU, DLMSPDU)	stop_timer() Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Dis
NC.T	time_out()	DLMSPDU=service_error(time-elapsed) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Locked

Tableau 9 *(fin)*

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
NC.T	(T_ABORT.ind(ErrorNb) & last_association()) | (exist_abort_ind() & not(last_association()))	DLMSPDU=service_error(provider-communication-error) Initiate.cnf(DLMSPDU) store_error(ErrorNb) inactivate_dlms_context()	Locked
Extract	Extract_OK	update_dlms_context(AppCtx, _) Initiate.cnf(DLMSPDU)	Idle
Extract	not(Extract_OK)	DLMSPDU=service_error(deciphering-error) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Dis
Idle	A_Disconnect.req()	T_ABORT.req(Strong=0) $purge()	Locked
Idle	Abort.req() active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(AbortReqAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) inactivate_dlms_context()	Locked
Idle	Initiate.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(AuthenticationReqAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) init_timer(T2)	AC.T
Idle	(T_ABORT.ind(ErrorNb) & last_association()) | (exist_abort_ind() & not(last_association()))	store_error(ErrorNb) inactivate_dlms_context()	Locked
Dis	last_association()	T_ABORT.req(Strong=0)	Locked
Dis	not(last_association())	$none()	Locked

Tableau 10 – Tableau de gestion des services confirmés côté Serveur

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Idle	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, ConfirmedReqAPSE)	Decipher_OK=decipher(APSEPDU, DLMSPDU)	Decipher
Decipher	Decipher_OK	ConfirmedService.ind(DLMSPDU)	CS.E
Decipher	not(Decipher_OK)	$none()	Locked
CS.E	ConfirmedService.rsp(DLMSPDU) & dlmspdu_type(DLMSPDU, confirmed-service) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(ConfirmedRespAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	Idle
CS.E	ConfirmedService.rsp(DLMSPDU) & dlmspdu_type(DLMSPDU, confirmed-error) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(ConfirmedErrorAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	Idle
CS.E	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked

Table 9 *(concluded)*

Initial state	Triggering condition	Set of actions	Final state
NC.T	(T_ABORT.ind(ErrorNb) & last_association()) | (exist_abort_ind() & not(last_association()))	DLMSPDU=service_error(provider-communication-error) Initiate.cnf(DLMSPDU) store_error(ErrorNb) inactivate_dlms_context()	Locked
Extract	Extract_OK	update_dlms_context(AppCtx, _) Initiate.cnf(DLMSPDU)	Idle
Extract	not(Extract_OK)	DLMSPDU=service_error(deciphering-error) Initiate.cnf(DLMSPDU) inactivate_dlms_context()	Dis
Idle	A_Disconnect.req()	T_ABORT.req(Strong=0) $purge()	Locked
Idle	Abort.req() active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(AbortReqAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) inactivate_dlms_context()	Locked
Idle	Initiate.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(Authentication ReqAPSE, _) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) init_timer(T2)	AC.T
Idle	(T_ABORT.ind(ErrorNb) & last_association()) | (exist_abort_ind() & not(last_association()))	store_error(ErrorNb) inactivate_dlms_context()	Locked
Dis	last_association()	T_ABORT.req(Strong=0)	Locked
Dis	not(last_association())	$none()	Locked

Table 10 – Server end confirmed service management table

Initial state	Triggering condition	Set of actions	Final state
Idle	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, ConfirmedReqAPSE)	Decipher_OK=decipher(APSEPDU, DLMSPDU)	Decipher
Decipher	Decipher_OK	ConfirmedService.ind(DLMSPDU)	CS.E
Decipher	not(Decipher_OK)	$none()	Locked
CS.E	ConfirmedService.rsp(DLMSPDU) & dlmspdu_type(DLMSPDU, confirmed-service) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(ConfirmedResp APSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	Idle
CS.E	ConfirmedService.rsp(DLMSPDU) & dlmspdu_type(DLMSPDU, confirmed-error) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(ConfirmedError APSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU)	Idle
CS.E	T_ABORT.ind(ErrorNb)	store_error(ErrorNb) $purge()	Locked

Tableau 11 – Tableau de gestion des services confirmés côté Client

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Idle	ConfirmedService.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(ConfirmedReqAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) init_timer(T2)	CC.T
CC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, ConfirmedRespAPSE)	Decipher_OK=decipher(APSEPDU, DLMSPDU)	Decipher
CC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, ConfirmedErrorAPSE) & convert(APSEPDU, DLMSPDU)	stop_timer() ConfirmedService.cnf(DLMSPDU)	Idle
CC.T	time_out()	DLMSPDU=service_error(time-elapsed) ConfirmedService.cnf(DLMSPDU)	Locked
CC.T	(T_ABORT.ind(ErrorNb) & last_association()) | (exist_abort_ind() & not(last_association()))	DLMSPDU=service_error(provider-communication-error) ConfirmedService.cnf(DLMSPDU) store_error(ErrorNb) inactivate_dlms_context()	Locked
Decipher	Decipher_OK	stop_timer() ConfirmedService.cnf(DLMSPDU)	Idle
Decipher	not(Decipher_OK)	stop_timer() DLMSPDU=service_error(deciphering-error) ConfirmedService.cnf(DLMSPDU)	Idle

Table 11 – Client end confirmed service management table

Initial state	Triggering condition	Set of actions	Final state
Idle	ConfirmedService.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP)	APSEPDU=build_tasepdu(ConfirmedReqAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=0, APSEPDU) Init_timer(T2)	CC.T
CC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, ConfirmedRespAPSE)	Decipher_OK=decipher(APSEPDU, DLMSPDU)	Decipher
CC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, ConfirmedErrorAPSE) & convert(APSEPDU, DLMSPDU)	Stop_timer() ConfirmedService.cnf(DLMSPDU)	Idle
CC.T	time_out()	DLMSPDU=service_error(time-elapsed) ConfirmedService.cnf(DLMSPDU)	Locked
CC.T	(T_ABORT.ind(ErrorNb) & last_association()) | (exist_abort_ind() & not(last_association()))	DLMSPDU=service_error(provider-communication-error) ConfirmedService.cnf(DLMSPDU) store_error(ErrorNb) inactivate_dlms_context()	Locked
Decipher	Decipher_OK	stop_timer() ConfirmedService.cnf(DLMSPDU)	Idle
Decipher	not(Decipher_OK)	stop_timer() DLMSPDU=service_error(deciphering-error) ConfirmedService.cnf(DLMSPDU)	Idle

Tableau 12 – Tableau de gestion des services non sollicités côté Serveur

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Idle	Unsolicited.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & check_client()	APSEPDU=build_tasepdu(UnsolicitedReqAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=1, APSEPDU) build_evt(InformationReported)	Idle
Idle	not(Unsolicited.req(_) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & check_client()) & (exist_unsolicited_req() & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & not(check_client()))	T_ABORT.req(Strong=0) $purge()	Locked
Locked	Unsolicited.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & address(CallingAddr, CalledList)	Connect_OK=server_connect(CallingAddr, CalledList)	Connect
Connect	Connect_OK	APSEPDU=build_tasepdu(UnsolicitedReqAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=1, APSEPDU) build_evt(InformationReported)	Alarm
Connect	not(Connect_OK)	build_evt(InformationNotReported)	Locked
Alarm	exist_unsolicited_req(Unsolicited.req (DLMSPDU)) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & check_client()	APSEPDU=build_tasepdu(UnsolicitedReqAPSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=1, APSEPDU) build_evt(InformationReported)	Alarm
Alarm	not(exist_unsolicited_req()) | exist_unsolicited_req() & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & not(check_client())	$none()	Dis
Dis	last_association()	T_ABORT.req(Strong=0)	Locked
Dis	not(last_association())	$none()	Locked

Table 12 – Server end unsolicited service management table

Initial state	Triggering condition	Set of actions	Final state
Idle	Unsolicited.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & check_client()	APSEPDU=build_tasepdu(UnsolicitedReq APSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=1, APSEPDU) build_evt(InformationReported)	Idle
Idle	not(Unsolicited.req(_) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & check_client()) & (exist_unsolicited_req() & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & not(check_client()))	T_ABORT.req(Strong=0) $purge()	Locked
Locked	Unsolicited.req(DLMSPDU) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & address(CallingAddr, CalledList)	Connect_OK=server_connect(CallingAddr, CalledList)	Connect
Connect	Connect_OK	APSEPDU=build_tasepdu(UnsolicitedReq APSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=1, APSEPDU) build_evt(InformationReported)	Alarm
Connect	not(Connect_OK)	build_evt(InformationNotReported)	Locked
Alarm	exist_unsolicited_req(Unsolicited.req (DLMSPDU)) & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & check_client()	APSEPDU=build_tasepdu(UnsolicitedReq APSE, DLMSPDU) T_DATA.req(STSAP, DTSAP, Pr=1, APSEPDU) build_evt(InformationReported)	Alarm
Alarm	not(exist_unsolicited_req()) | exist_unsolicited_req() & active_dlms_context(STSAP, DTSAP) & check_tsap(STSAP, DTSAP) & not(check_client())	$none()	Dis
Dis	last_association()	T_ABORT.req(Strong=0)	Locked
Dis	not(last_association())	$none()	Locked

Tableau 13 – Tableau de gestion des services non sollicités côté Client

Etat initial	Condition de déclenchement	Ensemble d'actions	Etat final
Idle	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	Unsolicited.ind(DLMSPDU)	Idle
Locked	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	set_dlms_context(ClientType) Unsolicited.ind(DLMSPDU)	Locked
AC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	stop_timer() Unsolicited.ind(DLMSPDU) init_timer(T2)	AC.T
NC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	stop_timer() Unsolicited.ind(DLMSPDU) init_timer(T2)	NC.T
CC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	stop_timer() Unsolicited.ind(DLMSPDU) init_timer(T2)	CC.T

Table 13 – Client end unsolicited service management table

Initial state	Triggering condition	Set of actions	Final state
Idle	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	Unsolicited.ind(DLMSPDU)	Idle
Locked	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	set_dlms_context(ClientType) Unsolicited.ind(DLMSPDU)	Locked
AC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	stop_timer() Unsolicited.ind(DLMSPDU) init_timer(T2)	AC.T
NC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	stop_timer() Unsolicited.ind(DLMSPDU) init_timer(T2)	NC.T
CC.T	T_DATA.ind(STSAP, DTSAP, APSEPDU) & check_tsap(STSAP, DTSAP) & tasepdu_type(APSEPDU, UnsolicitedReqAPSE) & extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	stop_timer() Unsolicited.ind(DLMSPDU) init_timer(T2)	CC.T

Tableau 14 – Signification des états mentionnés dans les tableaux précédents

Etat	Signification
Locked	Etat de démarrage commun au Client et au Serveur
AC.T (Authentication at the Client end & waiting under Timer)	Sous le contrôle du réveil MVRT, l'utilisateur DLMS Client attend une réponse d'authentification
IR.E (Initiate Request Expected)	Attente d'une requête d'Initiate du Client après l'envoi d'une réponse d'authentification du Serveur
NC.T (No Context & waiting under Timer)	Sous le contrôle du réveil MVRT, l'utilisateur DLMS Client attend une réponse d'initialisation
Con.E (Context Expected)	Attente d'une réponse de l'utilisateur DLMS Serveur après réception d'une requête d'Initiate du Client
CC.T (Confirmed service at the Client end & waiting under Timer)	Sous le contrôle du réveil MVRT, l'utilisateur DLMS Client attend une réponse de service confirmé
CS.E (Confirmed Service response Expected)	Attente d'une réponse de l'utilisateur DLMS Serveur après réception d'une requête de service confirmé du Client
Idle	La configuration du contexte DLMS est réussie (état opérationnel d'attente de requêtes ou d'indications de service)
D.Deciph	Sous-état destiné à s'interroger sur la valeur d'authentification reçue côté Client
Decipher	Sous-état destiné à s'interroger sur le résultat d'une opération de décryptage côté Client ou Serveur
Connect	Sous-état destiné à s'interroger sur le résultat d'une éventuelle connexion physique côté Client ou Serveur
Extract	Sous-état destiné à s'interroger sur le résultat d'une opération d'extraction et de décryptage côté Client ou Serveur et sur la valeur d'authentification reçue côté Serveur
Dis (Disconnection)	Sous-état destiné à s'interroger sur l'opportunité d'une éventuelle déconnexion physique
Alarm	Sous-état réservé au traitement des requêtes de services non sollicités en attente côté Serveur

Table 14 – Meanings of the states listed in the preceding tables

State	Meaning
Locked	Startup state common to the Client and the Server
AC.T (Authentication at the Client end & waiting under Timer)	Under the control of the wakeup MVRT, the Client DLMS user is waiting for an authentication response
IR.E (Initiate Request Expected)	Wait for an Initiate request from the Client after transmission of an authentication response by the Server
NC.T (No Context & waiting under Timer)	Under the control of the wakeup MVRT, the Client DLMS user is waiting for an initialization response
Con.E (Context Expected)	Wait for a response from the Server DLMS user after reception of an Initiate request from the Client
CC.T (Confirmed service at the Client end & waiting under Timer)	Under the control of the wakeup MVRT, the Client DLMS user is waiting for a confirmed service response
CS.E (Confirmed Service response Expected)	Wait for a response from the Server DLMS user after reception of a confirmed service request from the Client
Idle	DLMS context successfully configured (operational state waiting for service requests or indications)
D.Deciph	Sub-state intended to inquire about the authentication value received at the Client end
Decipher	Sub-state intended to inquire about the result of a deciphering operation at the Client end or Server end
Connect	Sub-state intended to inquire about the result of a possible physical connection at the Client end or Server end
Extract	Sub-state intended to inquire about the result of an extraction and deciphering operation at the Client end or Server end and about the authentication value received at the Server end
Dis (Disconnection)	Sub-state intended to inquire about the appropriateness of a possible physical disconnection
Alarm	Sub-state reserved for the processing of pending unsolicited service requests at the Server end

Tableau 15 – Définition des procédures et des fonctions classées par ordre alphabétique

Procédure ou fonction	Définition
active_dlms_context(STSAP, DTSAP)	Récupération du couple (STSAP, DTSAP) dans le contexte DLMS actif mis à disposition
address(CallingAddr, CalledAddr) ou address(CallingAddr, CalledList)	Mise à jour du contexte DLMS pour l'occurrence courante de l'automate d'application – cas du Client: récupération de l'adresse physique du Client et de l'adresse physique du Serveur – cas du Serveur: récupération de l'adresse physique du Serveur et des adresses physiques des Clients à partir de la variable d'administration ForalArmClientList
build_evt(InformationReported) ou build_evt(InformationNotReported) ou built_evt(AbortIndication)	Coté Serveur, génération d'un événement externe informant du résultat du traitement d'un message non sollicité ou de la fin d'une association d'application. Il convient de noter que la fonction $purge génère aussi des événements de type AbortIndication automatique-ment
build_tasepdu(APSEPDU_type, DLMSPDU)	Construction d'une APSEPDU de type APSEPDU_type après cryptage éventuel. Dans le cas particulier où le type est AuthenticationReqAPSE, AuthenticationRespAPSE, InitiateReqAPSE, InitiateRespAPSE ou UnsolicitedReqAPSE, la fonction récupère les informations manquantes dans le contexte DLMS actif. Lorsque le type est AuthenticationRespAPSE, il y a aussi mémorisation du nombre aléatoire généré pour le calcul de la clef dédiée
check_client()	Vérification que le type du Client contenu dans le contexte DLMS actif correspond au destinataire de l'alarme à délivrer. Dans le cas contraire, vérification que ce destinataire est référencé par la variable d'administration ForAlarmClientList en association avec l'adresse physique du Client courant
check_client_type(DTSAP, ClientType)	Pour un VDE repéré par son DTSAP, vérification de l'identifiant du type du Client à partir de la variable d'administration CallingIdentifierList
check_tsap(STSAP, DTSAP)	Vérification qu'un DTSAP est connu du Serveur et qu'un couple (STSAP, DTSAP) correspond à l'occurrence courante de l'automate d'application
client_connect(CallingAddr, CalledAddr)	La définition de cette fonction dépend du support de communication utilisé et doit donc être précisée dans chaque cas
convert(APSEPDU, DLMSPDU)	Extraction sans décryptage d'une DLMSPDU à partir d'une APSEPDU
d_decipher(AuthenticationRespAPSE, DKey)	Du côté Client, décryptage des valeurs cryptées contenues dans une AuthenticationRespAPSE, vérification de la valeur du nombre décrypté, puis calcul et mémorisation de la clef dédiée
decipher(APSEPDU, DLMSPDU)	Extraction et décryptage d'une DLMSPDU en fonction du mode de cryptage du contexte d'application
dlmspdu_type(DLMSPDU, initiate) ou dlmspdu_type(DLMSPDU, initiate-error) ou dlmspdu_type(DLMSPDU, confirmed-service) ou dlmspdu_type(DLMSPDU, confirmed-error)	Selon le cas, vérification qu'une DLMSPDU est de type: – InitiateResponse – ConfirmedServiceError sur erreur du service Initiate – ConfirmedServiceResponse, GetStatusResponse, GetNameListResponse, GetVariableAttributeResponse, ReadResponse ou WriteResponse – ConfirmedServiceError sur erreur d'un service autre que Initiate

Table 15 – Definition of the procedures and functions classified in alphabetical order

Procedure or function	Definition
active_dlms_context(STSAP, DTSAP)	Retrieval of the (STSAP, DTSAP) pair in the active DLMS context made available
address(CallingAddr, CalledAddr) or address(CallingAddr, CalledList)	Updating of the DLMS context for the current occurrence of the application controller – case of the Client: extraction of the Client physical address and extraction of the Server physical address – case of the Server: extraction of the Server physical address and extraction of the Client physical addresses from the management variable ForAlarmClientList
build_evt(InformationReported) or build_evt(InformationNotReported) or built_evt(AbortIndication)	At the Server end, generation of an external event reporting the result of the processing of an unsolicited message or of the end of an association application. It should be noted that the $purge function also generates events of the type AbortIndication automatically
build_tasepdu(APSEPDU_type, DLMSPDU)	Building of an APSEPDU of the type APSEPDU_type after ciphering, if any. In the particular case where the type is AuthenticationReqAPSE, AuthenticationRespAPSE, InitiateReqAPSE, InitiateRespAPSE or UnsolicitedReqAPSE, the function retrieves the missing information in the active DLMS context. If the type is AuthenticationRespAPSE, there is also a storing of the random number used for the generation of the dedicated key
check_client()	Check that the Client type contained in the active DLMS context corresponds to the addressee of the alarm to be delivered. In the opposite case, check that this addressee is included in the management variable ForAlarmClientList in association with the physical address of the current Client
check_client_type(DTSAP, ClientType)	For a VDE identified by its DTSAP, check of the identifier of the Client type from the management variable CallingIdentifierList
check_tsap(STSAP, DTSAP)	Check that a DTSAP is known by the Server and that a (STSAP, DTSAP) pair corresponds to the current occurrence of the application controller
client_connect(CallingAddr, CalledAddr)	The definition of this function depends on the communication medium used and shall therefore be clarified in each case
convert(APSEPDU, DLMSPDU)	Extraction without deciphering of a DLMSPDU from an APSEPDU
d_decipher(AuthenticationRespAPSE, DKey)	At the Client end, deciphering of values contained in an AuthenticationRespAPSE, check of the deciphered number value, then generation and storing of the dedicated key
decipher(APSEPDU, DLMSPDU)	Extraction and deciphering of a DLMSPDU according to the ciphering mode of the application context
dlmspdu_type(DLMSPDU, initiate) or dlmspdu_type(DLMSPDU, initiate-error) or dlmspdu_type(DLMSPDU, confirmed-service) or dlmspdu_type(DLMSPDU, confirmed-error)	Check that a DLMSPDU is one of the type: – InitiateResponse – ConfirmedServiceError on an Initiate service error – ConfirmedServiceResponse, GetStatusResponse, GetNameListResponse, GetVariableAttributeResponse, ReadResponse or WriteResponse – ConfirmedServiceError on another service than Initiate error

Tableau 15 *(fin)*

Procédure ou fonction	Définition
exist_abort_ind()	Vérification de l'existence d'un événement de type T_ABORT.ind(ErrorNb)
exist_unsolicited_req() ou exist_unsolicited_req(Unsolicited.req(DLMSPDU))	Vérification de l'existence d'un événement de type Unsolicited.req(DLMSPDU) ou consommation d'un événement de type Unsolicited.req(DLMSPDU)
extract_authentication_req(APSEPDU, ClientType)	Du coté Serveur, à partir d'une APSEPDU de type AuthenticationReqAPSE, extraction du type du Client et de la valeur cryptée, et décryptage de cette dernière
extract_initiate_req(APSEPDU, AppCtx, CallingAddr, DLMSPDU)	Du coté Serveur, à partir d'une APSEPDU de type InitiateReqAPSE, extraction, décryptage et vérification de la valeur du nombre décrypté, du contexte d'application proposé AppCtx, de l'adresse physique du Client Appelant CallingAddr et d'une DLMSPDU avec décryptage en fonction du mode de cryptage du contexte d'application
extract_initiate_resp(APSEPDU, AppCtx, DLMSPDU)	A partir d'une APSEPDU de type InitiateRespAPSE, extraction du contexte d'application négocié AppCtx et d'une DLMSPDU avec décryptage en fonction du mode de cryptage du contexte d'application
extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	A partir d'une APSEPDU de type UnsolicitedReqAPSE, extraction du type du Client cible et d'une DLMSPDU avec décryptage en fonction du mode de cryptage du contexte d'application
inactivate_dlms_context()	Désactivation du contexte DLMS actif correspondant à l'occurrence courante de l'automate d'application
init_timer(T2)	Armement du réveil MVRT avec la valeur de la variable T2
last_association()	Vérification que l'association d'application courante est bien la dernière
server_connect(CallingAddr, CalledList)	La définition de cette fonction dépend du support de communication utilisé et doit donc être précisée dans chaque cas
service_error(application-reference-type) ou service_error(initiate-type)	Construction de la DLMSPDU de type ConfirmedServiceError qui caractérise une erreur repérée au niveau application-reference ou initiate du type ServiceError (voir CEI 61334-4-41)
set_dlms_context(AppCtx, STSAP, DTSAP, ClientType, CallingAddr) ou set_dlms_context(ClientType)	Mise à jour et activation du contexte DLMS pour l'occurrence courante de l'automate d'application – cas du Serveur: si le contexte d'application proposé AppCtx n'appartient pas à la variable d'administration ApplicationContextNameList, c'est la valeur 0 qui est utilisée. Les autres informations nécessaires sont extraites de la variable d'administration ConfidentialItem – cas du Client: seul le type du Client est remonté
stop_timer()	Désarmement du réveil MVRT
store_error(ErrorNb)	Conservation du dernier numéro d'erreur fatale ErrorNb dans la variable d'administration FatalError
tasepdu_type(APSEPDU, APSEPDU_type)	Vérification qu'une APSEPDU est de type APSEPDU_type
time_out()	Déclenchement du réveil MVRT
update_dlms_context(AppCtx, DKey)	Mise à jour du contexte DLMS actif du Client pour l'occurrence courante de l'automate d'application avec la valeur du contexte d'application négocié AppCtx et de la clef dédiée DKey

Table 15 *(concluded)*

Procedure or function	Definition
exist_abort_ind()	Check of the existence of an event of the type T_ABORT.ind(ErrorNb)
exist_unsolicited_req() or exist_unsolicited_req(Unsolicited.req (DLMSPDU))	Check of the existence of an event of the type Unsolicited.req(DLMSPDU) or consumption of an event of the type Unsolicited.req(DLMSPDU)
extract_authentication_req(APSEPDU, ClientType)	At the Server end, from an APSEPDU of AuthenticationReqAPSE type, extraction of Client type and ciphered value, then deciphering of this value
extract_initiate_req(APSEPDU, AppCtx, CallingAddr, DLMSPDU)	At the Server end, from an APSEPDU of the type InitiateReqAPSE, extraction, deciphering and check of the value of the deciphered number, of the proposed application context AppCtx, of the physical address of the Caller Client CallingAddr and of a DLMSPDU with deciphering according to the ciphering mode of the application context
extract_initiate_resp(APSEPDU, AppCtx, DLMSPDU)	From an APSEPDU of the type InitiateRespAPSE, extraction of the negotiated application context AppCtx and of a DLMSPDU with deciphering according to the ciphering mode of the application context
extract_unsolicited_req(APSEPDU, ClientType, DLMSPDU)	From an APSEPDU of the type UnsolicitedReqAPSE, extraction of the target Client type and of a DLMSPDU with deciphering according to the ciphering mode of the application context
inactivate_dlms_context()	Inactivation of the active DLMS context corresponding to the current occurrence of the application controller
init_timer(T2)	Setting of the wakeup MVRT with the value of the variable T2
last_association()	Check that the current application association is the last one
server_connect(CallingAddr, CalledList)	The definition of this function depends on the communication medium used and shall therefore be clarified in each case
service_error(application-reference-type) or service_error(initiate-type)	Building of the DLMSPDU of the ConfirmedServiceError type that characterizes a ServiceError type error detected at the application-reference or initiate level (see IEC 61334-4-41)
set_dlms_context(AppCtx, STSAP, DTSAP, ClientType, CallingAddr) or set_dlms_context(ClientType)	Updating and activation of the DLMS context for the current occurrence of the application controller – case of the Server: if the proposed application context AppCtx does not belong to the management variable ApplicationContextNameList, the value 0 is used. The other necessary information is extracted from the management variable ConfidentialItem – case of the Client: only the Client type is sent up
stop_timer()	Stopping of the wakeup MVRT
store_error(ErrorNb)	Storing of the last fatal error number ErrorNb in the management variable FatalError
tasepdu_type(APSEPDU, APSEPDU_type)	Check that an APSEPDU is of the type APSEPDU_type
time_out()	Triggering of the wakeup MVRT
update_dlms_context(AppCtx, DKey)	Updating of the active DLMS context of the Client for the current occurrence of the application controller with the value of the negotiated application context AppCtx and of the dedicated key Dkey

4.13 Répertoire et traitement des erreurs

Les erreurs sont répertoriées selon le codage suivant:

EA erreur de la sous-couche Application

– séparateur

N numéro de l'erreur

F erreur fatale

Tableau 16 – Tableau récapitulatif des erreurs

EA-1	APSEPDU incorrecte (erreur de codage ASN.1)
	Cette erreur conduit à simplement ignorer le message
EA-2	Erreur d'identification du Serveur ou type du Client non autorisé
	Cette erreur ne peut se produire que côté Serveur. Elle conduit à générer une confirmation négative de service
EA-3	Erreur d'adressage (DTSAP inconnu)
	Cette erreur ne peut se produire que côté Serveur. Elle conduit à générer une confirmation négative de service
EA-4	Erreur de décryptage côté Client
	Cette erreur conduit à générer une confirmation négative de service
EA-5	Erreur de décryptage côté Serveur
	Cette erreur conduit à simplement ignorer le message
EA-6	Expiration du délai T2 sans qu'aucune PDU correcte n'ait été reçue
	Cette erreur ne peut se produire que côté Client. Elle conduit à générer une confirmation négative de service

Les erreurs non fatales ne sont pas signalées.

4.13 List and processing of errors

The errors are listed with the following codes:

EA error in the Application sub-layer

– separator

N number of the error

F fatal error

Table 16 – Error summary table

EA-1	APSEPDU incorrect (ASN.1 encoding error)
	This error simply causes the message to be ignored
EA-2	Server identification error or Client type not authorized
	This error can only occur at the Server end. It leads to the generation of a negative service confirmation
EA-3	Addressing error (unknown DTSAP)
	This error can only occur at the Server end. It leads to the generation of a negative service confirmation
EA-4	Deciphering error at the Client end
	This error leads to the generation of a negative service confirmation
EA-5	Deciphering error at the Server end
	This error simply causes the message to be ignored
EA-6	Expiry of the period T2 without any correct PDU being received
	This error can only occur at the Client end. It leads to the generation of a negative service confirmation

Non-fatal errors are not reported.

Annexe A
(normative)

Langage de spécification

A.1 Vocabulaire et règles de fonctionnement

Pour décrire sans ambiguïté le rôle de chaque sous-couche, la spécification utilise un formalisme en tableaux modélisant le comportement réel par un automate à nombre fini d'états.

A chaque automate correspond un unique tableau logique qui peut éventuellement être représenté sous la forme de plusieurs tableaux physiques. Ce découpage se justifie lorsque le tableau logique est particulièrement important.

A chaque occurrence d'automate correspond une instance (copie active distincte) du tableau logique de l'automate de référence.

Chaque tableau physique est composé de lignes appelées lignes d'état. Chaque ligne d'état décrit la condition de déclenchement (colonne 2) pour que la machine passe d'un état initial (colonne 1) à un état final (colonne 4) en exécutant un ensemble d'actions (colonne 3).

Le premier état initial est l'état de démarrage de l'automate. Cet état est unique; il est particularisé au moyen de l'attribut souligné.

Un état d'arrêt de l'automate est un état final pour lequel aucune ligne d'état n'est définie avec cet état comme état initial. Un automate est infini lorsqu'il ne possède aucun état d'arrêt. Un automate fini peut posséder un ou plusieurs états d'arrêt. Ces états sont également représentés avec l'attribut souligné. Compte tenu de cette convention, l'ordre dans lequel sont présentés les états dans un tableau physique n'a aucune importance.

Cette même règle s'applique lorsque plusieurs lignes d'état référencent le même état initial car les conditions de déclenchement sont toujours exclusives les unes des autres. L'ordre des lignes d'un tableau physique n'est donc guidé que par de simples considérations de présentation. Il est cependant logique de commencer par décrire les transitions de l'état de démarrage.

Un ensemble d'actions d'une ligne d'état doit être considéré comme une section critique (c'est-à-dire une séquence non interruptible). Les actions qui y sont décrites doivent être exécutées dans l'ordre séquentiel où elles sont écrites. Une action est définie par un appel à une procédure nommée instanciée avec une liste de zéro, un ou plusieurs paramètres entre parenthèses. Toute procédure nommée référencée doit faire l'objet d'une description séparée. Il existe cependant deux actions prédéfinies: l'affectation = et l'action vide $none() (absence d'action).

La condition de déclenchement associée à une ligne d'état peut éventuellement être composée de plusieurs sous-conditions. L'évaluation d'une condition de déclenchement composée passe toujours par l'évaluation de toutes les sous-conditions qu'elle contient. Ainsi, l'ordre d'écriture des sous-conditions est sans importance.

Les opérateurs supportés pour exprimer une condition composée sont, d'une part, les opérateurs logiques & (et logique), | (ou logique), not() (non logique) et, d'autre part, les opérateurs de comparaison (<, >, ≤, ≥, = et <>).

Annex A
(normative)

Specification language

A.1 Vocabulary and operating rules

To describe the role of each sub-layer unambiguously, the specification uses a table formalism modelling the real behaviour by a controller with a finite number of states.

To each controller corresponds a unique logic table; this logic table may be broken down into several physical tables, if it is particularly large.

To each controller occurrence corresponds an instance (distinct active copy) of the logic table of the reference controller.

Each physical table consists of lines known as state lines. Each state line describes the triggering condition (column 2) for the machine to pass from an initial state (column 1) to a final state (column 4) by executing a set of actions (column 3).

The first initial state is the startup state of the controller. This state is unique; it is particularized by means of the underlined attribute.

A stop state of the controller is a final state for which no state line is defined with this state as initial state. A controller is infinite when it does not have a stop state. A finite controller may have one or more stop states. These states are also represented with the attribute underlined. This convention means that the order in which the states are presented in a physical table is not important.

The same rule applies when several state lines refer to the same initial state, because the triggering conditions are always mutually exclusive. The order of the lines in a physical table is therefore governed by presentation considerations only. Nevertheless, it is logical to begin by describing the transitions of the startup state.

A set of actions in a state line shall be considered as a critical section (i.e. an uninterruptible sequence). The actions described there shall be executed in the order in which they are written. An action is defined by an invocation of a named procedure instantiated with a zero list, one or more parameters between parentheses. All referenced named procedures shall be the subject of separate descriptions. However, there are two predefined actions: assignment = and empty action $none() (no action).

The triggering condition associated with a state line may be composed of several sub-conditions. The assessment of a composite triggering condition always involves the assessment of all the sub-conditions that it contains. Therefore, the order in which the sub-conditions are written is unimportant.

The operators supported for expressing composite conditions are the logic operators & (logic and), | (logic or), not() (logic no), and the comparison operators (<, >, ≤, ≥, = and <>).

Il existe deux types de condition de déclenchement.

Une condition de type simple est, par définition, évaluée instantanément. Elle peut éventuellement être composée mais, dans ce cas, toutes les sous-conditions sont de type simple. Une fonction nommée booléenne est un exemple de condition de type simple. Toute fonction nommée booléenne référencée doit faire l'objet d'une description séparée.

Une condition de type événementiel exprime l'attente d'un événement. Elle peut éventuellement être composée de plusieurs sous-conditions événementielles ou simples.

Lorsque l'évaluation d'une condition de déclenchement conduit à un résultat vrai, la condition se trouve réalisée. La réalisation d'une condition de déclenchement conduit toujours à une transition d'état.

Un événement peut être défini comme étant un élément contribuant à la réalisation d'une condition de déclenchement de type événementiel.

Lorsque un événement est inclus dans une condition de déclenchement de type événementiel qui se trouve réalisée, il est automatiquement consommé. Un événement ne peut être consommé qu'une seule fois.

Tout événement survenant lorsque l'occurrence d'automate qui est susceptible de le consommer se trouve dans un état où cette consommation est impossible est stocké chronologiquement dans une zone appelée file inter-automate.

Ainsi, chaque automate dispose d'une unique file qu'il partage entre ses propres occurrences d'automate. La taille de cette file est supposée quasi infinie; son organisation et sa gestion ne sont pas décrites ici. Toutefois, il convient de noter qu'une purge partielle de la file (c'est-à-dire liée aux seuls événements concernant l'occurrence d'automate courante) est automatiquement effectuée pour toute transition d'état partant de l'état de démarrage.

Il doit exister également un mécanisme d'auto-purge conduisant à la suppression automatique des événements entrants et manifestement non consommables. En outre, il existe une procédure nommée prédéfinie $purge() qui correspond à l'action de purge totale de la file inter-automate courante. Toutes les occurrences de l'automate correspondant se retrouvent alors dans l'état de démarrage.

La production d'événements est assurée par certaines des actions décrites dans un ensemble d'actions associé à une ligne d'état. Un événement interne ne peut être consommé que par l'automate qui l'a produit. Un événement externe est toujours consommé par un autre automate que celui qui l'a produit.

Il convient de noter que l'absence d'un événement (exprimé par une sous-condition de type événementiel encapsulée dans l'opérateur logique not()) est toujours une sous-condition de type simple.

Lorsque, pour un état initial, il existe une ligne d'état où la condition de déclenchement est d'un certain type (simple ou événementiel), alors toutes les lignes d'état ayant le même état initial doivent posséder des conditions de déclenchement du même type.

Lorsque ce type est simple, l'état initial est appelé sous-état. Un sous-état est particularisé au moyen de l'attribut italique. Il est transitoire et peut toujours être éliminé; sa présence dans un tableau physique n'est justifiée que par un souci de clarté de la présentation. Dans le cas particulier d'un sous-état de démarrage, une condition particulière a été prédéfinie; il s'agit de la condition $true(), qui est toujours vraie.

There are two types of triggering conditions.

A simple condition is, by definition, assessed instantaneously. It may be composite but, in such cases, all the sub-conditions shall be of the simple type. A boolean named function is an example of a simple condition. All referenced boolean named functions shall be the subject of separate descriptions.

An event condition expresses the wait for an event. It may be composed of several event or simple sub-conditions.

When the assessment of a triggering condition gives a true result, the condition is satisfied. The satisfaction of a triggering condition always leads to a state transition.

An event can be defined as an element contributing to the satisfaction of an event-type triggering condition.

When an event is included in an event-type triggering condition which is satisfied, it is automatically consumed. An event can be consumed only once.

Any event that occurs when the controller occurrence that is likely to consume it is in a state where this consumption is impossible is stored chronologically in an area known as the inter-controller queue.

Each controller thus has a single queue that it shares among its own controller occurrences. The size of this queue is assumed to be quasi-infinite; its organization and its management are not described here. However, it should be noted that a partial purge of the queue (i.e. related only to the events concerning the current controller occurrence) is automatically carried out for any state transition starting from the startup state.

There must also be a self-purge mechanism for automatic deletion of incoming events that are manifestly not consumable. Moreover, there is a predefined named procedure $purge() which corresponds to the action of total purge of the current inter-controller queue. All the occurrences of the corresponding controller then return to the startup state.

Events are produced by some of the described actions in a set of actions associated with a state line. An internal event can be consumed only by the controller that has produced it. An external event is always consumed by a controller other than the one that has produced it.

It should be noted that the absence of an event (expressed by an event sub-condition encapsulated in the logic operator not()) is always a simple sub-condition.

When, for an initial state, there is a state line where the triggering condition is of a certain type (simple or event), then all the state lines having the same initial state shall have triggering conditions of the same type.

When this type is simple, the initial state is called sub-state. A sub-state is particularized by means of the italic attribute. It is transient and can always be removed; its presence in a physical table is justified only by improved clarity of presentation. In the special case of a startup sub-state, a special condition, $true(), has been predefined, which is always true.

Enfin, les variables référencées dans les conditions de déclenchement et les actions décrites dans un tableau physique restent locales à chaque occurrence d'automate. Il existe également une variable prédéfinie (la variable non liée _) destinée à remplacer tout paramètre inexploité dans n'importe quelle fonction ou procédure nommée.

A.2 Entity et Entity Invocation

Il est intéressant d'effectuer un parallèle entre les éléments du langage de spécification présenté ici et certains concepts développés par l'OSI (Open Systems Interconnection).

Pour chaque sous-couche, on note, par exemple, que la notion d'Entity correspond à un automate, alors que le terme Entity Invocation est similaire à l'expression occurrence d'automate.

The variables referred to in the triggering conditions and the actions described in a physical table remain local with respect to each controller occurrence. There is also a predefined variable (the unlinked variable _) intended to replace any unused parameter in any function or named procedure.

A.2 Entity and Entity Invocation

It is interesting to draw a parallel between the elements of the specification language described here and certain concepts developed by the OSI (Open Systems Interconnection).

For each sub-layer, for example, the notion of Entity corresponds to a controller, whereas the term Entity Invocation is similar to controller occurrence.

Annexe B
(normative)

Liste des erreurs fatales

Toute occurrence de l'une des erreurs fatales répertoriées en provenance des couches basses est remontée localement à la sous-couche Application, qui la stocke dans une variable DLMS d'administration. Cette variable, de nom FatalError, est accessible par l'intermédiaire du Serveur DLMS d'administration des protocoles de communication.

Tableau B.1 – Numéros des erreurs de FatalError

Numéro	1	2
Erreur	ET-1F	ET-2F

Quelle que soit la sous-couche incriminée, l'occurrence d'une erreur fatale conduit à

– éventuellement, faire s'arrêter la sous-couche inférieure au moyen de la primitive de service abort.req,

– éventuellement, informer la sous-couche supérieure au moyen de la primitive abort.ind avec, comme paramètre, le numéro de l'erreur fatale,

– effectuer une réinitialisation totale de l'occurrence d'automate correspondante.

Annex B
(normative)

List of fatal errors

Any occurrence of one of the listed fatal errors coming from the low layers is sent up locally to the Application sub-layer, which stores it in a management DLMS variable, FatalError. This variable is accessible through the communication protocols management DLMS Server.

Table B.1 – FatalError error numbers

Number	1	2
Error	ET-1F	ET-2F

Whatever sub-layer is involved, the occurrence of a fatal error results in

– if appropriate, stopping of the next lower sub-layer by means of the service primitive abort.req,

– if appropriate, informing of the next higher sub-layer by means of the primitive abort.ind with the number of the fatal error as parameter,

– complete reinitialization of the corresponding controller occurrence.

Annexe C
(normative)

Authentification et nombres aléatoires

Les algorithmes qui peuvent être utilisés pour l'authentification nécessitent la génération et la transformation de nombres aléatoires de 64 bits.

C.1 Génération de nombres aléatoires

La méthode choisie est basée sur un décalage suivi d'une addition (modulo 2) sur une suite de bits correctement initialisée.

Le registre R0 contient la valeur d'initialisation K qui est additionnée (modulo 2) à deux index de 64 bits. Le résultat est ensuite rangé dans le registre R2. Les index doivent être choisis de manière à posséder un comportement pseudo-aléatoire.

Le registre R1 contient les coefficients d'un polynôme générateur (modulo 2) irréductible et primitif de degré 63.

Grâce à la fonction f, les registres R1 et R2 sont ensuite combinés de la manière suivante:

– si le bit de poids fort de R2 est égal à 1, alors R2 est décalé d'un bit vers les bits de poids fort et est additionné (modulo 2) à R1;
– sinon, R2 est décalé d'un bit vers les bits de poids fort.

On obtient ainsi le premier nombre aléatoire NA(1) utilisé par un appareil lors du premier échange effectué. Les nombres aléatoires suivants NA(i) sont obtenus par le même processus en remplaçant la valeur d'initialisation K par le nombre précédemment calculé NA(i–1):

$$\begin{cases} NA(i) = f\,((K \oplus (INDEX1, INDEX2)), R1) & \text{pour } i = 1 \\ NA(i) = f\,((NA(i-1) \oplus (INDEX1, INDEX2)), R1) & \text{pour } i > 1 \end{cases}$$

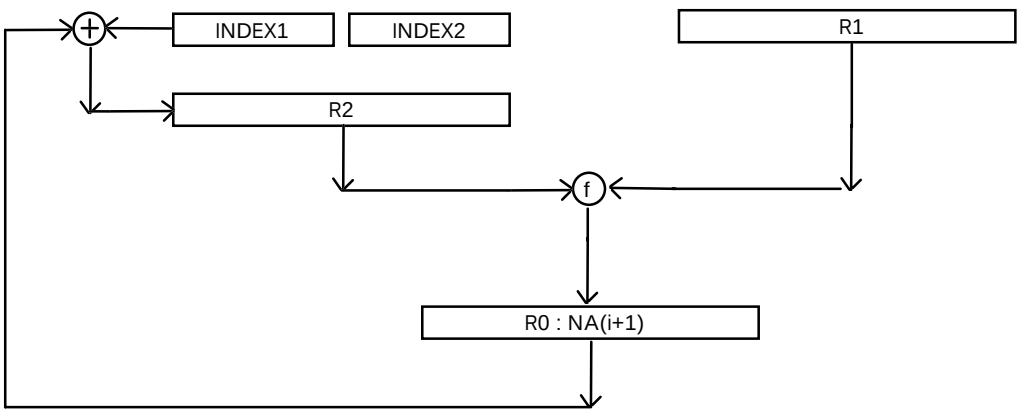

Figure C.1 – Schéma de réalisation

Annex C
(normative)

Authentication and random numbers

The algorithms that may be used for authentication require the generation and processing of 64 bit random numbers.

C.1 Random number generation

The chosen method is based on a shift followed by an addition (modulo 2) on a properly initialized bit string.

The R0 register contains the K initialization value which is added (modulo 2) to two 64 bit indexes. The result is then stored in the R2 register. The indexes shall be chosen so that they have a pseudo-random behaviour.

The R1 register contains the factors of an irreducible and primitive generator polynomial (modulo 2) of degree 63.

Thanks to the f function, the R1 and R2 registers are then combined as follows:

– if the R2 most significant bit equals 1, R2 is shifted by one bit towards the most significant bits and is added (modulo 2) to R1;
– otherwise, R2 is shifted by one bit towards the most significant bits.

Hence a first random number generated NA(1) is obtained and used by a device during the first exchange. The following random numbers NA(i) are obtained using the same procedure by replacing the K initialization value by the previous random number generated NA(i–1):

$$\begin{cases} NA(i) = f((K \oplus (INDEX1, INDEX2)), R1) & \text{for } i = 1 \\ NA(i) = f((NA(i-1) \oplus (INDEX1, INDEX2)), R1) & \text{for } i > 1 \end{cases}$$

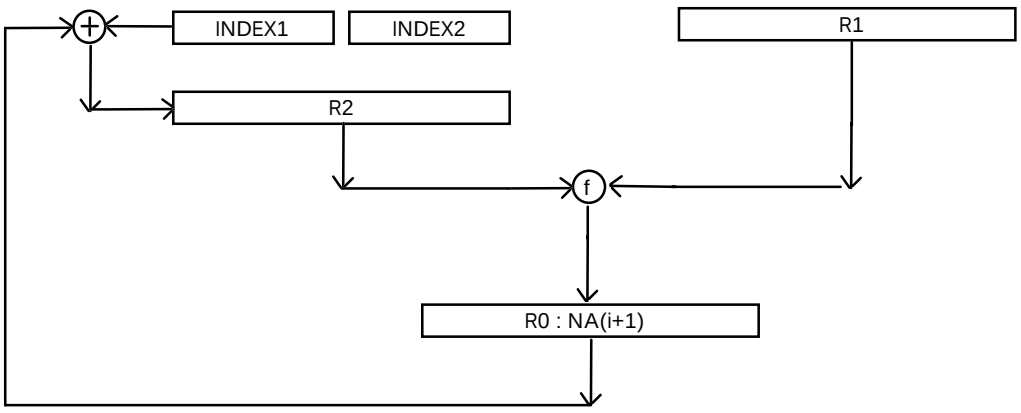

Figure C.1 – Realization diagram

C.2 Paramètres de fonctionnement

La valeur d'initialisation de R0 est fixée à 00 01 02 03 04 05 06 07 en hexadécimal.

Le polynome générateur choisi est $P(x) = x^{63} + x^{1} + 1$. Le registre R1 est donc chargé avec la valeur 80 00 00 00 00 00 00 03 en hexadécimal.

Du côté Serveur, Index1 est pris égal à la valeur du champ modification-count de la variable d'administration ModificationCount pour chaque association d'application. Par symétrie, côté Client, Index1 peut être un registre interne incrémenté chaque fois qu'une confirmation de service confirmé est reçue.

Index2 est pris égal au nombre de tops d'horloge sur l'équipement considéré.

C.3 Transformation des nombres aléatoires

Pour l’algorithme DES

La transformation des nombres aléatoires est définie comme une permutation circulaire vers la gauche de 8 des 64 bits de chaque nombre.

Dans le cas particulier où un nombre est composé d'octets tous identiques, la transformation devient invariante et une nouvelle valeur aléatoire doit être tirée.

Pour l’algorithme SHA

La transformation des nombres aléatoires se fait par concaténation de la clef d'authentification à la fin de la chaîne correspondant au nombre aléatoire pour former un message de 16 octets. L’empreinte SHA de ce message est ensuite calculée et le nombre crypté transmis est obtenu par extraction de 8 octets sur les 20 que comporte l'empreinte, en prenant un octet sur deux (le premier, le troisième et ainsi de suite).

C.2 Operating parameters

The initial value of R0 is set to 00 01 02 03 04 05 06 07 in hexa.

The chosen generator polynomial is $P(x) = x^{63} + x^{1} + 1$. The R1 register is therefore loaded with the value 80 00 00 00 00 00 00 03 in hexa.

At the Server end, Index1 is set to the value of the modification-count field of the ModificationCount management variable for each application association. By symmetry, at the Client end, Index1 can be an internal register incremented at each received confirmation of confirmed service.

Index2 is set to the timer top number of the equipment.

C.3 Random number transformation

For DES algorithm

The random number transformation is defined by a left circular shift of 8 of the 64 bits of each number.

In the particular case of a number which is composed of identical octets, the transformation becomes invariant and a new random value has to be set.

For SHA algorithm

The random number transformation consists in concatenation of the authentication key at the end of the string corresponding to the random number to form a 16 octet message. The SHA print of this message is then computed and the ciphered number transmitted is obtained by extracting 8 octets from this 20 octets print, taking one octet in two (the first, the third and so on).

Annexe D
(normative)

Algorithme de brouillage pour la confidentialité des données

L'algorithme présenté dans la présente annexe fait partie intégrante du contexte d'application par défaut du protocole Application+.

D.1 Confidentialité des données

La confidentialité des données est assurée par une fonction de brouillage et de débrouillage des messages échangés au niveau de la sous-couche Application.

En effet, compte tenu de la sécurité déjà obtenue par le mécanisme d'authentification, l'utilisation du même algorithme dans ce cadre conduirait à des échanges inutilement longs, surtout avec de grands messages.

D.2 Séquence binaire de longueur maximale

Pour garantir une sécurité suffisante, le brouillage doit être une fonction du message et du temps. En pratique, l'algorithme à retenir peut se contenter d'une fonction pseudo-aléatoire de période longue, obtenue, par exemple, à partir du concept de séquence binaire de longueur maximale dont voici un bref rappel.

Soit A et B deux suites de n bits représentées par les conventions suivantes:

$A = (a_1, a_2 \ldots a_n)$

$B = (b_1, b_2 \ldots b_n)$

On peut définir une suite $A' = (a'_1, a'_2 \ldots a'_n)$ comme étant le résultat d'une fonction de décalage F_n définie par l'équation $A' = F_n(A, B)$ avec

$a'_i = a_{i+1}$	pour toute valeur de i de 1 à n–1
$a'_n = a_1b_1 \oplus a_2b_2 \oplus \ldots \oplus a_nb_n$	($\oplus$ désignant l'addition modulo 2)

D.3 Fonction de brouillage et de débrouillage

Soit MO le message origine et MC le message après brouillage. En adoptant un découpage en k tranches de n bits (la dernière tranche possèdant l bits, $1 \leq l \leq n$), ces messages peuvent être représentés avec le formalisme suivant:

$MO = (MO_1, MO_2 \ldots MO_k)$	
avec $MO_i = (mo^i_1, mo^i_2 \ldots mo^i_n)$	pour toute valeur de i de 1 à k–1
et $MO_k = (mo^i_1, mo^i_2 \ldots mo^i_l)$	$1 \leq l \leq n$
$MC = (MC_1, MC_2 \ldots MC_k)$	
avec $MC_i = (mc^i_1, mc^i_2 \ldots mc^i_n)$	pour toute valeur de i de 1 à k–1
et $MC_k = (mo^i_1, mo^i_2 \ldots mo^i_l)$	$1 \leq l \leq n$

Annex D
(normative)

Masking algorithm for data confidentiality

The algorithm described in this annex is an integral part of the default application context of the Application+ protocol.

D.1 Data confidentiality

The confidentiality of data is provided by a masking and unmasking function on the messages exchanged at the Application sub-layer.

Indeed, considering the security obtained by the authentication mechanism, the use of the same algorithm in this case would lead to unnecessarily long exchanges, especially with long messages.

D.2 Maximum length binary sequence

In order to guarantee adequate security, the masking shall be a function of the message and of the time. In practice, it is sufficient for the algorithm used to have a long period pseudo-random function, obtained, for example, from the maximum length binary sequence concept summarized below.

Let A and B be two strings of n bits represented by the following conventions:

$A = (a_1, a_2 \ldots a_n)$

$B = (b_1, b_2 \ldots b_n)$

A string $A' = (a'_1, a'_2 \ldots a'_n)$ can be defined as the result of a shift function F_n defined by the equation $A' = F_n(A, B)$ where

$a'_i = a_{i+1}$	for all values of i from 1 to n–1
$a'_n = a_1b_1 \oplus a_2b_2 \oplus \ldots \oplus a_nb_n$	($\oplus$ designating modulo 2 addition)

D.3 Masking and unmasking function

Let MO be the source message and MC the message after masking. By adopting a breakdown into k sections of n bits (the last section owning l bits, $1 \leq l \leq n$), these messages can be represented with the following formalism:

$MO = (MO_1, MO_2 \ldots MO_k)$	
where $MO_i = (mo^i_1, mo^i_2 \ldots mo^i_n)$	for all values of i from 1 to k–1
and $MO_k = (mo^i_1, mo^i_2 \ldots mo^i_l)$	$1 \leq l \leq n$
$MC = (MC_1, MC_2 \ldots MC_k)$	
where $MC_i = (mc^i_1, mc^i_2 \ldots mc^i_n)$	for all values of i from 1 to k–1
and $MC_k = (mo^i_1, mo^i_2 \ldots mo^i_l)$	$1 \leq l \leq n$

La fonction F_c de brouillage qui définit la relation MC = F_c(MO) est telle que

$MC_j = R_j \oplus Mo_j$ pour toute valeur de j de 1 à k ($\oplus$ désignant l'addition vectorielle modulo 2)

$R_j = F_n(R_{j-1}, B)$ pour toute valeur de j de 2 à k

R_1 est la clef de brouillage initiale de l'algorithme.

Le détail du calcul de la clef de brouillage R_1 est le suivant:

– pour l'algorithme DES:
  – R_1 = n bits de poids faible ($n \leq 64$) du nombre aléatoire Ns généré par le Serveur lors de la phase d'authentification;
  – si R_1 = "0...0"H, alors R_1 = "F...F"H.
– pour l'algorithme SHA:
  – R_1 = les deux derniers octets de l'empreinte SHA générée par le Serveur lors de la phase d'authentification.

Enfin, puisque l'addition modulo 2 est involutive, on constate que la fonction F_d de débrouillage est identique à la fonction de brouillage:

$$MO = F_d(MC) = F_d(F_c(MO)) = F_c(F_c(MO))$$

D.4 Paramètres de fonctionnement

Le choix des paramètres n et B est un problème de mise en oeuvre devant respecter la clause de confidentialité.

The masking function F_c which defines the relationship $MC = F_c(MO)$ is such that

$MC_j = R_j \oplus Mo_j$ for all values of j from 1 to k
($\oplus$ designating modulo 2 vector addition)

$R_j = F_n(R_{j-1}, B)$ for all values of j from 2 to k

R_1 is the initial masking key of the algorithm.

The masking key R_1 is calculated as follows:

– for DES algorithm:
  – R_1 = n least significant bits ($n \leq 64$) of the random number Ns generated by the Server during the authentication step;
  – if R_1 = "0...0"H, then R_1 = "F...F"H.
– for SHA algorithm:
  – R_1 = the last two octets of the SHA print generated by the Server during the authentication step.

As modulo 2 addition is involutive, the unmasking function F_d is identical to the masking function:

$$MO = F_d(MC) = F_d(F_c(MO)) = F_c(F_c(MO))$$

D.4 Operating parameters

The choice of n and B parameters is an implementation issue that has to refer to confidentiality.

Annexe E
(normative)

Identifiants et mode de brouillage

E.1 Identifiant Client et identifiant Serveur

Pour un équipement Serveur, un Client est identifiable par son type:

Client-type::=INTEGER(-32 768..32 767)

Poids forts	Poids faibles
Identificateur (13 bits)	Classe de l'objet (3 bits)

La valeur de la classe de l'objet est choisie à 111 pour correspondre à l'objet VAA.

Les valeurs de l'idendificateur sont exprimées sur 13 bits. Ces valeurs sont attribuées librement.

Une variable DLMS d'administration fournit la liste des types de Client admissibles pour chaque VDE d'un équipement réel. Cette variable, de nom CallingIdentifierList, est accessible par l'intermédiaire du Serveur DLMS d'administration des protocoles de communication.

Un Serveur s'identifie grâce à l'attribut Serial Number défini par DLMS (voir CEI 61334-4-41) comme étant l'identifiant unique du VDE dans le monde:

Server-identifier::=OCTET STRING

Une variable DLMS d'administration fournit l'identifiant Serveur de chaque VDE d'un équipement réel. Cette variable, de nom ApplicationList, est accessible par l'intermédiaire du Serveur DLMS d'administration des protocoles de communication.

E.2 Mode de brouillage

Le mode de brouillage utilisé pour la confidentialité des données est contenu dans le contexte d'application. La règle de calcul de ce mode est précisée dans le tableau suivant.

Tableau E.1 – Mode de brouillage

Classes de service DLMS	Types de Client d'administration [1)]	Autres types de Client
ConfirmedServiceRequest	Brouillage avec la clef dédiée	Brouillage avec la clef dédiée
ReadRequest	Brouillage avec la clef dédiée	Brouillage avec la clef dédiée
ReadResponse	Brouillage avec la clef dédiée	Pas de brouillage
WriteRequest	Brouillage avec la clef dédiée	Brouillage avec la clef dédiée
Autres classes de service	Pas de brouillage	Pas de brouillage
[1)] C'est-à-dire ayant accès au Serveur DLMS d'administration des protocoles de communication.		

Annex E
(normative)

Identifiers and masking mode

E.1 Client and Server identifiers

For a Server equipment, a Client can be identified by its type:

Client-type::=INTEGER(-32 768..32 767)

Most significant bit	Least significant bit
Item identifier (13 bits)	Object class (3 bits)

The object class value is chosen as 111 for the VAA object.

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

A management DLMS variable provides the list of the Client types available for each VDE of a real equipment. This variable, CallingIdentifierList, is accessible through the communication protocols management DLMS Server.

A Server identifies itself with the Serial Number attribute which is defined by DLMS (see IEC 61334-4-41) as the single identifier of the VDE throughout the world:

Server-identifier::=OCTET STRING

A management DLMS variable provides the Server identifier of each VDE of a real equipment. This variable, ApplicationList, is accessible through the communication protocols management DLMS Server.

E.2 Masking mode

The masking mode used for data confidentiality is contained in the application context. The generation rule of this mode is described in the following table.

Table E.1 – Masking mode

DLMS service classes	Management Client type [1)]	Other Client types
ConfirmedServiceRequest	Masking using dedicated key	Masking using dedicated key
ReadRequest	Masking using dedicated key	Masking using dedicated key
ReadResponse	Masking using dedicated key	No masking
WriteRequest	Masking using dedicated key	Masking using dedicated key
Other service classes	No masking	No masking
[1)] That is to say accessing to the communication protocol management DLMS Server.		

Standards Survey

The IEC would like to offer you the best quality standards possible. To make sure that we continue to meet your needs, your feedback is essential. Would you please take a minute to answer the questions overleaf and fax them to us at +41 22 919 03 00 or mail them to the address below. Thank you!

Customer Service Centre (CSC)

International Electrotechnical Commission
3, rue de Varembé
1211 Genève 20
Switzerland

or

Fax to: **IEC**/CSC at +41 22 919 03 00

Thank you for your contribution to the standards-making process.

Nicht frankieren
Ne pas affranchir

Non affrancare
No stamp required

A Prioritaire

RÉPONSE PAYÉE

SUISSE

Customer Service Centre (CSC)
International Electrotechnical Commission
3, rue de Varembé
1211 GENEVA 20
Switzerland

Q1 Please report on **ONE STANDARD** and **ONE STANDARD ONLY**. Enter the exact number of the standard: *(e.g. 60601-1-1)*

.....

Q2 Please tell us in what capacity(ies) you bought the standard *(tick all that apply).* I am the/a:

purchasing agent ❑
librarian ❑
researcher ❑
design engineer ❑
safety engineer ❑
testing engineer ❑
marketing specialist ❑
other.....

Q3 I work for/in/as a: *(tick all that apply)*

manufacturing ❑
consultant ❑
government ❑
test/certification facility ❑
public utility ❑
education ❑
military ❑
other.....

Q4 This standard will be used for: *(tick all that apply)*

general reference ❑
product research ❑
product design/development ❑
specifications ❑
tenders ❑
quality assessment ❑
certification ❑
technical documentation ❑
thesis ❑
manufacturing ❑
other.....

Q5 This standard meets my needs: *(tick one)*

not at all ❑
nearly ❑
fairly well ❑
exactly ❑

Q6 If you ticked NOT AT ALL in Question 5 the reason is: *(tick all that apply)*

standard is out of date ❑
standard is incomplete ❑
standard is too academic ❑
standard is too superficial ❑
title is misleading ❑
I made the wrong choice ❑
other .....

Q7 Please assess the standard in the following categories, using the numbers:
(1) unacceptable,
(2) below average,
(3) average,
(4) above average,
(5) exceptional,
(6) not applicable

timeliness.....
quality of writing.....
technical contents.....
logic of arrangement of contents .....
tables, charts, graphs, figures.....
other .....

Q8 I read/use the: *(tick one)*

French text only ❑
English text only ❑
both English and French texts ❑

Q9 Please share any comment on any aspect of the IEC that you would like us to know:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Enquête sur les normes

La CEI ambitionne de vous offrir les meilleures normes possibles. Pour nous assurer que nous continuons à répondre à votre attente, nous avons besoin de quelques renseignements de votre part. Nous vous demandons simplement de consacrer un instant pour répondre au questionnaire ci-après et de nous le retourner par fax au +41 22 919 03 00 ou par courrier à l'adresse ci-dessous. Merci !

Centre du Service Clientèle (CSC)

Commission Electrotechnique Internationale
3, rue de Varembé
1211 Genève 20
Suisse

ou

Télécopie: **CEI**/CSC +41 22 919 03 00

Nous vous remercions de la contribution que vous voudrez bien apporter ainsi à la Normalisation Internationale.

Nicht frankieren
Ne pas affranchir

Non affrancare
No stamp required

A Prioritaire

RÉPONSE PAYÉE

SUISSE

Centre du Service Clientèle (CSC)
Commission Electrotechnique Internationale
3, rue de Varembé
1211 GENÈVE 20
Suisse

Q1 Veuillez ne mentionner qu'**UNE SEULE NORME** et indiquer son numéro exact: (*ex. 60601-1-1)*

.....

Q2 En tant qu'acheteur de cette norme, quelle est votre fonction?
(cochez tout ce qui convient)
Je suis le/un:

agent d'un service d'achat ❑
bibliothécaire ❑
chercheur ❑
ingénieur concepteur ❑
ingénieur sécurité ❑
ingénieur d'essais ❑
spécialiste en marketing ❑
autre(s).....

Q3 Je travaille:
(cochez tout ce qui convient)

dans l'industrie ❑
comme consultant ❑
pour un gouvernement ❑
pour un organisme d'essais/ certification ❑
dans un service public ❑
dans l'enseignement ❑
comme militaire ❑
autre(s).....

Q4 Cette norme sera utilisée pour/comme
(cochez tout ce qui convient)

ouvrage de référence ❑
une recherche de produit ❑
une étude/développement de produit ❑
des spécifications ❑
des soumissions ❑
une évaluation de la qualité ❑
une certification ❑
une documentation technique ❑
une thèse ❑
la fabrication ❑
autre(s).....

Q5 Cette norme répond-elle à vos besoins:
(une seule réponse)

pas du tout ❑
à peu près ❑
assez bien ❑
parfaitement ❑

Q6 Si vous avez répondu PAS DU TOUT à Q5, c'est pour la/les raison(s) suivantes:
(cochez tout ce qui convient)

la norme a besoin d'être révisée ❑
la norme est incomplète ❑
la norme est trop théorique ❑
la norme est trop superficielle ❑
le titre est équivoque ❑
je n'ai pas fait le bon choix ❑
autre(s) .....

Q7 Veuillez évaluer chacun des critères ci-dessous en utilisant les chiffres
(1) inacceptable,
(2) au-dessous de la moyenne,
(3) moyen,
(4) au-dessus de la moyenne,
(5) exceptionnel,
(6) sans objet

publication en temps opportun .....
qualité de la rédaction.....
contenu technique .....
disposition logique du contenu .....
tableaux, diagrammes, graphiques, figures.....
autre(s) .....

Q8 Je lis/utilise: *(une seule réponse)*

uniquement le texte français ❑
uniquement le texte anglais ❑
les textes anglais et français ❑

Q9 Veuillez nous faire part de vos observations éventuelles sur la CEI:

.....
.....
.....
.....
.....
.....

ISBN 2-8318-4546-7

ICS 91.140.50; 33.040.40

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND