NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
61334-6

Première édition
First edition
2000-06

Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs –

Partie 6: Règles d'encodage A-XDR

Distribution automation using distribution line carrier systems –

Part 6: A-XDR encoding rule

Numéro de référence
Reference number
CEI/IEC 61334-6:2000

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.

NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC
61334-6

Première édition
First edition
2000-06

Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs –

Partie 6: Règles d'encodage A-XDR

Distribution automation using distribution line carrier systems –

Part 6: A-XDR encoding rule

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 W

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

SOMMAIRE

Pages

AVANT-PROPOS 4
INTRODUCTION 6

Articles

1 Domaine d'application et objet 8
2 Références normatives 8
3 Caractéristiques générales d'A-XDR 10
4 Structure d'un codage 10
5 Règles pour le codage 16
5.1 Le champ Identificateur 16
5.2 Le champ Longueur 18
5.3 Le champ Contenu 18
6 Procédures de codage 20
6.1 Codage d'une valeur INTEGER 20
6.2 Codage de la valeur BOOLEAN 26
6.3 Codage d'une valeur ENUMERATED 28
6.4 Codage d'une valeur BIT STRING 28
6.5 Codage d'une valeur OCTET STRING 30
6.6 Codage de la valeur CHOICE 34
6.7 Types étiquetés (étiquetage implicite, explicite et explicite ASN.1) 36
6.8 Composants OPTIONAL et DEFAULT 40
6.9 Codage d'une valeur SEQUENCE 42
6.10 Codage d'une valeur SEQUENCE OF 44
6.11 Codage du type VisibleString 48
6.12 Codage du type de GeneralizedTime 50
6.13 Codage de la valeur/du type NULL ASN.1 50

Annexe A (informative) Extensibilité 52
Annexe B (informative) Types et mots-clefs de l'ASN.1 utilisés en DLMS 54
Annexe C (informative) Exemples de codage A-XDR pour les PDU DLMS 56

Figure 1 – Structure de base de l'encodage BER 10
Figure 2 – Structure d'un codage BER construit 12
Figure 3 – Structure d'un codage A-XDR construit 12
Figure 4 – Structure du codage d'un nombre entier de longueur variable 24

CONTENTS

Page

FOREWORD 5
INTRODUCTION 7

Clause

1 Scope and object 9
2 Normative references 9
3 General characteristics of A-XDR 11
4 Structure of an encoding 11
5 Rules for encoding 17
5.1 The Identifier field 17
5.2 The Length field 19
5.3 The Contents field 19
6 Encoding procedures 21
6.1 Encoding of an INTEGER value 21
6.2 Encoding of a BOOLEAN value 27
6.3 Encoding of an ENUMERATED value 29
6.4 Encoding of a BIT STRING value 29
6.5 Encoding of an BYTE STRING value 31
6.6 Encoding of a CHOICE value 35
6.7 Tagged types (implicit, explicit and ASN.1 explicit tagging) 37
6.8 OPTIONAL and DEFAULT components 41
6.9 Encoding of a SEQUENCE value 43
6.10 Encoding of a SEQUENCE OF value 45
6.11 Encoding of the VisibleString type 49
6.12 Encoding of the GeneralizedTime type 51
6.13 Encoding of the ASN.1 NULL type/value 51

Annex A (informative) Extensibility 53
Annex B (informative) ASN.1 types and keywords used in DLMS 55
Annex C (informative) Examples of A-XDR encoding for DLMS PDUs 57

Figure 1 – The basic BER structure 11
Figure 2 – The structure of a constructed BER encoding 13
Figure 3 – The structure of a constructed A-XDR encoding 13
Figure 4 – Structure of the variable-length integer encoding 25

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

AUTOMATISATION DE LA DISTRIBUTION À L'AIDE DE SYSTÈMES DE COMMUNICATION À COURANTS PORTEURS –

Partie 6: Règles d'encodage A-XDR

AVANT-PROPOS

1) La CEI (Commission Électrotechnique 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, spécifications techniques, 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 norme internationale CEI 61334-6 a été préparé par le comité technique 57 de la CEI: Conduite des systèmes de puissance et communications associées.

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
57/451/FDIS	57/474/RVD

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

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 3.

Les annexes A, B et C sont données uniquement à titre d'information.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant 2003. A cette date, la publication sera

- reconduite;
- supprimée;
- remplacée par une édition révisée, ou
- amendée.

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

DISTRIBUTION AUTOMATION USING DISTRIBUTION LINE CARRIER SYSTEMS –

Part 6: A-XDR encoding rule

FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.
3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.
6) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. The IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61334-6 has been prepared by IEC technical committee 57: Power system control and associated communications.

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

FDIS	Report on voting
57/451/FDIS	57/474/RVD

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

This publication has been drafted in accordance with the ISO/IEC Directives, Part 3.

Annexes A, B and C are for information only.

The committee has decided that the contents of this publication will remain unchanged until 2003. At this date, the publication will be

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

INTRODUCTION

La Recommandation UIT-T X.208 spécifie un langage formel (ASN.1 = notation de syntaxe abstraite numéro Un) permettant aux spécifications de la couche d'application de définir les types[1)] d'information qu'elles souhaitent échanger. Une représentation de cette information peut être obtenue en appliquant un ensemble de règles de codage aux valeurs des types définis à l'aide de la notation ASN.1. L'application de ces règles de codage donne une syntaxe de transfert pour ces valeurs.

Bien qu'un grand nombre d'ensembles de règles de codage puissent être imaginés, pendant longtemps, seul un ensemble – l'encodage BER = Basic Encoding Rules (règles de codage de base) – a été normalisé (voir la Recommandation UIT-T X.209). C'est surtout dû au fait qu'il est relativement bien adapté à une large gamme d'applications. Cependant, dans certains cas particuliers, l'encodage BER peut présenter clairement certaines redondances. Eviter ces redondances en fournissant des règles de codage alternatives pour ce type de cas est le but de nouvelles normes de syntaxes de transfert récemment développées (DER, CER, PER). En fait, le but n'est pas de fournir des alternatives générales à l'encodage BER mais plutôt des alternatives spécialisées, qui soient plus appropriées que l'encodage BER dans des cas particuliers.

Contrairement à ces règles de codage pour une utilisation générale, la présente norme spécifie un nouvel ensemble de règles de codage pour une utilisation spécifique – A-XDR – qui s'adapte le mieux au contexte DLMS (voir CEI 61334-4-41). L'objectif principal est de coder les PDU (unités de données du protocole) DLMS de telle sorte que le nombre des octets et la complexité de codage/décodage – longueur du code nécessaire, performance et temps de traitement – des PDU soient optimisés[2)]. Cet objectif est atteint par deux principes de base:

a) A-XDR spécifie des règles de codage seulement pour une sous-ensemble de types ASN.1: il s'agit du sous-ensemble utilisé pour la spécification DLMS (c'est la raison pour laquelle A-XDR est prévue pour des utilisations spécifiques).

b) A-XDR spécifie des règles de codage orientées octets.

---

1) L'ASN.1 spécifie également une notation pour la spécification de la valeur d'un type défini.

2) Lorsque l'on considère seulement la taille des PDU, le PER est plus performant qu'A-XDR. Cependant, cette meilleure performance de compactage – objectif principal de PER – est obtenue par une utilisation beaucoup plus importante de champs de bits au lieu de champs d'octets pour coder des valeurs différentes. Pour réduire encore la taille des codes, la variante la plus complexe des PER (le PER non aligné) bénéficie également de la limitation de valeurs de types contraints. On gagne ainsi en compacité au détriment du temps de calcul. De plus, les deux variantes de PER (alignée et non alignée) sont incompatibles, et il est recommandé que les applications supportent les deux variantes. Cette complexité signifie que PER n'est pas optimal pour le contexte DLMS. Les règles de codage «plus légères» d'A-XDR sont plus adaptées à cet environnement simple, qui est dans certains cas très pauvre en ressources.

INTRODUCTION

ITU-T Recommendation X.208 specifies a formal language (ASN.1 = Abstract Syntax Notation One) which enables application layer specifications to define the types[1)] of information they need to exchange. A representation of this information can be derived by applying a set of encoding rules to values of types defined using the ASN.1 notation. Application of these encoding rules produces a transfer syntax for such values.

Although many such sets of encoding rules could be imagined, for a long time only one single set – the BER = Basic Encoding Rules – has been standardized (see ITU-T Recommendation X.209). This is mainly because BER is quite adequate for a wide range of applications. On the other hand, in some particular cases, BER can obviously be redundant. Avoiding this redundancy by providing alternative encoding rules for those particular cases is the scope of some recently developed new transfer syntax standards (DER, CER, PER). Clearly, the aim is not to provide general-purpose, but rather specialized, alternatives to the BER, which are more suitable than the BER in some respects.

Contrary to these general-purpose encoding rules, this standard specifies a new, special-purpose set of encoding rules – A-XDR – which fits in best with the DLMS context (see IEC 61334-4-41). The principal objective is to encode DLMS PDUs in such a way that the PDUs byte count and encoding/decoding complexity – the length of the required code, its processing performance and time – are optimized[2)]. This objective is fulfilled by two basic principles.

a) A-XDR specifies encoding rules only for a subset of ASN.1 types: for the subset which is used for the DLMS specification. (That is why A-XDR is special-purpose.)

b) A-XDR specifies byte-oriented encoding rules.

---

1) ASN.1 also specifies a notation for the specification of the value of a defined type.

2) With respect to the PDU size only, PER over-performs A-XDR. However, this better compacting performance – the principal objective of PER – is achieved by a much more extensive use of bit fields instead of byte fields to encode different values. To reduce encoding sizes further, the more complex PER variant (the Unaligned PER) also benefits from the limitation of values of constrained types. Gain on compactness is thus obtained at the expense of computational overhead. Furthermore, PER comes with two, incompatible variants (Aligned and Unaligned), and it is recommended that implementations should support both of them. This complexity means that PER is not optimal for the DLMS context. The 'lighter-weight' A-XDR encoding rules are more suitable to that simple environment, which is in some cases very poor in resources.

AUTOMATISATION DE LA DISTRIBUTION À L'AIDE DE SYSTÈMES DE COMMUNICATION À COURANTS PORTEURS –

Partie 6: Règles d'encodage A-XDR

1 Domaine d'application et objet

La présente partie de la CEI 61334 définit un ensemble de règles de codage – les règles de codages A-XDR[3)] – susceptibles d'être utilisées pour obtenir la spécification d'une syntaxe de transfert pour les valeurs de types définis dans la norme principale DLMS à l'aide de la notation ASN.1 (voir la CEI 61334-4-41). Ces règles de codage A-XDR doivent également être appliquées pour le décodage de cette syntaxe de transfert afin d'identifier les valeurs de données transférées.

Les règles de codage A-XDR:

- sont utilisées au moment de la communication;
- fournissent un codage optimal[4)] pour les PDU DLMS.

NOTE Si A-XDR réussit à assurer un codage optimal pour les PDU DLMS, il sera utilisé comme règle de codage par défaut pour les protocoles de communication basés DLMS. Néanmoins, les règles de codage par défaut – ainsi que les règles optionnelles pouvant finalement être utilisées – seront spécifiées dans le document Couche d'Application du protocole donné (par exemple la CEI 61334-4-42), comme une partie du contexte d'Application.

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 la présente partie de la CEI 61334. Pour les références datées, les amendements ultérieurs ou les révisions de ces publications ne s'appliquent pas. Toutefois, les parties prenantes aux accords fondés sur la présente partie de la CEI 61334 sont invitées à rechercher la possibilité d'appliquer les éditions les plus récentes des documents normatifs indiqués ci-après. Pour les références non datées, la dernière édition du document normatif en référence s'applique. Les membres de l'ISO et de la CEI 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*

CEI 61334-4-42:1996, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4: Protocoles de communication de données – Section 42: Protocoles d'application – Couche application*

ISO/CEI 8825-2:1997, *Technologie de l'information – Règles d'encodage ASN.1: Spécification des règles de codage condensées (PER)*

Recommandation UIT-T X.208:1988, *Spécification de la syntaxe abstraite numéro UN (ASN.1)*

Recommandation UIT-T X.209:1988, *Spécification des règles de codage de base pour la notation de syntaxe abstraite numéro un (ASN.1)*

———————

3) A-XDR signifie Adapted XDR (XDR adapté) – en fait, ces règles de codage sont déduites d'une norme confirmée et existante Unix, nommée XDR (eXternal Data Representation = représentation externe des données, rfc 1014).

4) Voir note de bas de page 2 dans l'introduction.

DISTRIBUTION AUTOMATION USING DISTRIBUTION LINE CARRIER SYSTEMS –

Part 6: A-XDR encoding rule

1 Scope and object

This part of IEC 61334 defines a set of encoding rules – the A-XDR[3)] encoding rules – that may be used to derive the specification of a transfer syntax for values of types defined in the DLMS core standard using the ASN.1 notation (see IEC 61334-4-41). These A-XDR encoding rules are also to be applied for decoding such a transfer syntax in order to identify the data values being transferred.

The A-XDR encoding rules

- are used at the time of communication;
- provide optimal[4)] encoding for DLMS PDUs.

NOTE Provided that A-XDR ensures optimal encoding for DLMS PDUs, it is intended to be the default encoding rule for DLMS-based communication protocols. Nevertheless, the default – and also the possibly usable optional – encoding rules will be specified in the Application Layer document of the given protocol (for example, IEC 61334-4-42), as part of the Application context.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 61334. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 61334 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

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

IEC 61334-4-42:1996, *Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 42: Application protocols – Application layer*

ISO/IEC 8825-2:1997, *Information technology – ASN.1 Encoding rules: Specification of packed encoding rules (PER)*

ITU-T Recommendation X.208:1988, *Specification of Abstract Syntax Notation One (ASN.1)*

ITU-T Recommendation X.209:1988, *Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)*

---

3) A-XDR stands for Adapted XDR. In fact, these encoding rules are derived from a proven and *de facto* standard of the Unix world, called XDR (eXternal Data Representation, rfc1014).

4) See footnote 2 of the introduction.

3 Caractéristiques générales d'A-XDR

A-XDR spécifie des règles de codage pouvant être utilisées pour coder et décoder les valeurs d'une syntaxe abstraite définie comme les valeurs d'un seul type ASN.1 (le type principal). Ce type ASN.1 est soit un type simple soit un type composé. Un élément d'un type composé peut être un type simple ou un type composé lui-même.

Les règles de codage A-XDR exploitent le fait que l'émetteur et le récepteur d'une PDU DLMS utilisent exactement la même spécification de syntaxe abstraite. Alors qu'avec l'encodage BER, le codage de chaque valeur de tout type d'une syntaxe abstraite est réalisé en style type-longueur-valeur (TLV), A-XDR code le type et la longueur de la valeur seulement lorsque cette information est nécessaire. Ceci implique que, sans connaître ce type de valeur codée, il n'est pas possible de déterminer la structure du codage.

NOTE Cette méthode de codage a pour effet que les règles de codage A-XDR ne sont pas extensibles (voir annexe A).

Afin qu'A-XDR reste aussi simple que possible, certaines restrictions s'appliquent selon la syntaxe abstraite à coder, comme suit:

- aucun support de codage n'est fourni pour les types ASN.1 non utilisés en DLMS[5)];
- il convient que le type CHOICE ASN.1 contienne seulement des composants explicitement[6)] étiquetés.

A-XDR spécifie des règles de codage orientées octets. Ceci signifie que chaque partie du codage – et donc également le codage dans son ensemble – est un nombre entier d'octets.

4 Structure d'un codage

La base du codage BER (voir la Recommandation UIT-T X.209) est une structure composée de trois parties: type, longueur et valeur comme indiqué à la figure 1. Pour le BER, ces trois parties sont désignées par identificateur (I), longueur (L) et contenu (C). La partie identificateur identifie le type, la partie longueur permet de repérer la fin du contenu[7)] et la partie contenu véhicule l'une des valeurs possibles de ce type.

IEC 730/2000

Figure 1 – Structure de base de l'encodage BER

Le champ du contenu peut être simplement une série d'octets[8)] – codage primitif – ou une série de codages emboîtés – codage construit – comme indiqué à la figure 2.

_____

5) L'annexe B énumère les types et les mots-clefs ASN.1 utilisés dans la spécification DLMS.

6) Les termes «étiquetage explicite» et «étiquetage implicite» ont une signification légèrement différente pour l'A-XDR que celle spécifiée pour l'ASN.1 et les BER. Le paragraphe 6.7 traite de ces notions et introduit également le nouveau terme «explicitement étiqueté ASN.1».

7) En fait, pour les BER, le champ de longueur ne représente pas toujours littéralement la longueur du contenu. Les BER spécifient deux formes – définie et indéfinie – du champ de longueur. Même si le champ de longueur représente effectivement le nombre d'octets dans le champ de contenu lorsque la forme définie est utilisée, pour la forme indéfinie, le champ de longueur indique que le contenu est terminé par des octets fin-de-contenu.

8) Zéro ou plus.

3 General characteristics of A-XDR

A-XDR specifies encoding rules which can be used to encode and decode the values of an abstract syntax defined as the values of a single ASN.1 type (the outermost type). This single ASN.1 type is either a simple type or a composite type. A component of a composite type may be a simple type or a composite type itself.

The A-XDR encoding rules exploit the fact that the sender and the receiver of a DLMS PDU are operating exactly the same specification of the abstract syntax. While with BER the encoding of every value of any type of abstract syntax is constructed in type-length-value (TLV) style, A-XDR encodes the type and the length of the value only when this information is necessary. This implies that without knowledge of the type of value encoded it is not possible to determine the structure of the encoding.

NOTE This encoding method gives the result that A-XDR encoding rules are not extensible (see annex A).

In order to keep A-XDR as simple as possible, some restrictions apply with respect to the abstract syntax to be encoded as follows:

- no encoding support is provided for ASN.1 types which are not used in DLMS[5];
- the CHOICE ASN.1 type should contain only explicitly[6] tagged components.

A-XDR specifies byte-oriented encoding rules. This means that each part of the encoding – and therefore also the encoding of the whole – is an integral number of bytes.

4 Structure of an encoding

The basis of BER encoding (see ITU-T Recommendation X.209) is a structure, made up of three parts: type, length and value, as shown in figure 1. In BER, these three parts are termed identifier (I), length (L) and contents (C). The identifier part identifies the type, the length part allows the end of the contents[7] to be found, and the contents part conveys one of the possible values of that type.

IEC 730/2000

Figure 1 – The basic BER structure

The contents field can be simply a series of bytes[8] (primitive encoding) or a series of nested encoding (constructed encoding), as shown in figure 2.

5) Annex B enumerates the ASN.1 types and keywords which are used in the DLMS specification.

6) The terms "explicit tagging" and "implicit tagging" have a slightly different meaning in A-XDR than that specified for ASN.1 and BER. Subclause 6.7 deals with these notions and also introduces the new "ASN.1 explicit tagging" term.

7) In fact, for BER, the length field does not always literally represent the length of the contents. BER specifies two forms (definite and indefinite) of the length field. Although, when the definite form is used, the length field effectively represents the number of bytes in the contents field, for the indefinite form the length field indicates that the contents are terminated by end-of-contents bytes.

8) Zero or more.

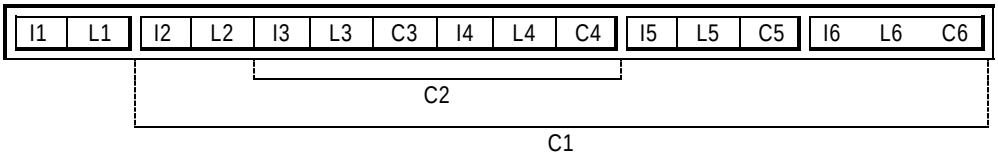


Figure 2 – Structure d'un codage BER construit

Cet emboîtement peut être aussi profond que nécessaire et s'arrête avec un codage primitif ou avec un codage construit ayant un contenu vide.

A-XDR est basée sur la même structure de codage, mais pour bénéficier du fait que l'émetteur et le récepteur d'une PDU DLMS utilisent exactement la même spécification de syntaxe abstraite, A-XDR ne code pas les champs Identificateur (I) et/ou Longueur (L) lorsque ces champs véhiculent une information redondante (le fait de ne pas coder l'un ou les deux de ces champs rend le codage ambigu). Un codage A-XDR construit peut donc être de structure similaire à celle illustrée à la figure 3.

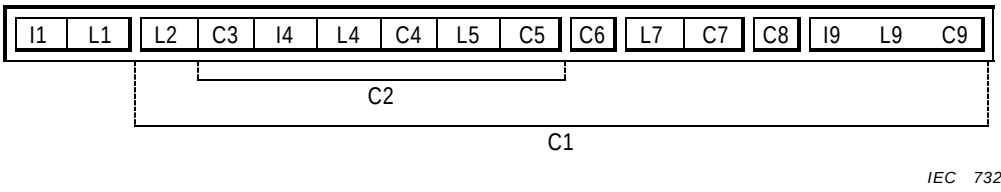


Figure 3 – Structure d'un codage A-XDR construit

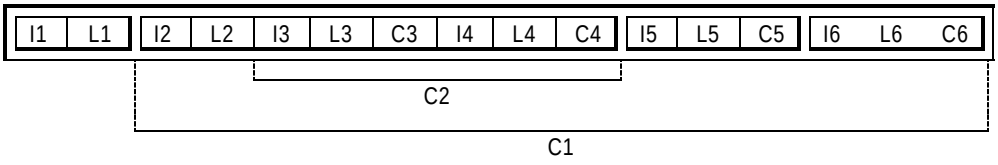


Figure 2 – The structure of a constructed BER encoding

The nesting can be as deep as needed and stops either with a primitive encoding or with a constructed encoding with empty contents.

A-XDR is based upon the same encoding structure, but in order to benefit from the fact that the sender and the receiver of a DLMS PDU are operating exactly the same specification of the abstract syntax, A-XDR does not encode the Identifier (I) and/or the Length (L) fields when those fields convey redundant information (when not to encode one or both of these fields does not result in uninterpretable, ambiguous encoding). A constructed A-XDR encoding therefore results in a structure as shown in figure 3.

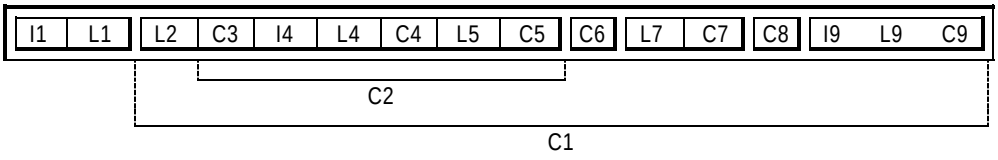


Figure 3 – The structure of a constructed A-XDR encoding

Les règles de codage A-XDR spécifient:

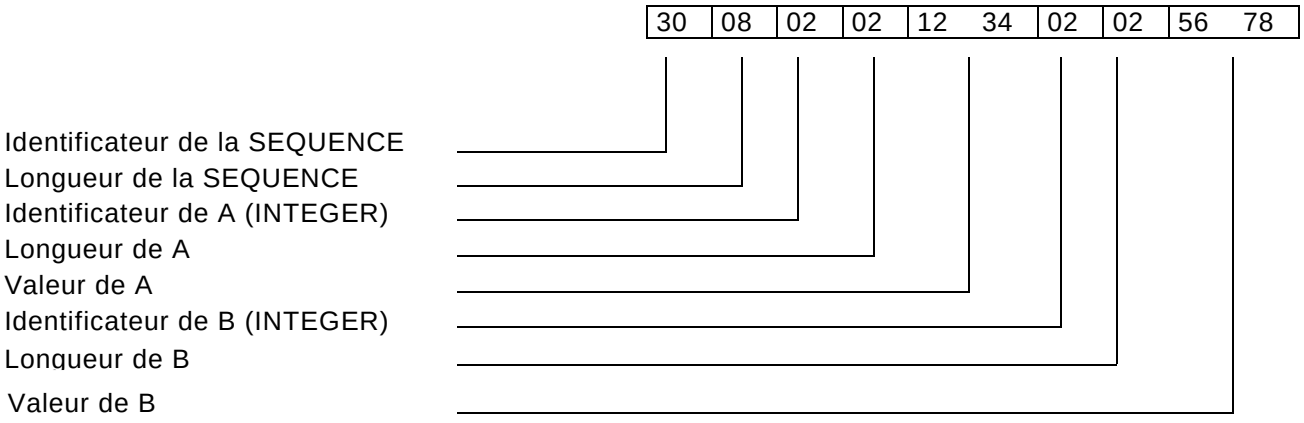
- des règles de codage pour les champs de contenu;
- les conditions pour lesquelles il convient que le champ L soit présent, et le codage de ce champ;
- les conditions pour lesquelles il convient que le champ I soit présent, et le codage de ce champ.

Exemple

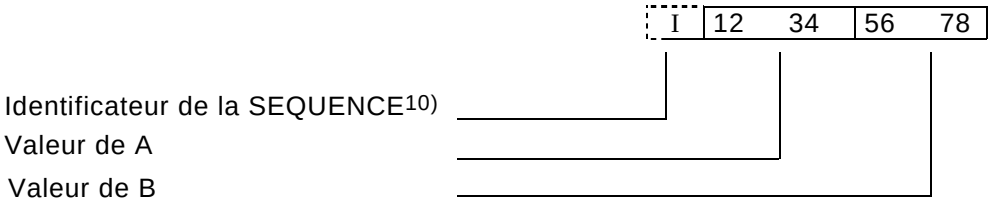
En considérant le type composé ASN.1 suivant:

```
valeur ::= SEQUENCE {
        A       Integer16,
        B       Unsigned16
    }
Integer16 ::= INTEGER(–32768..32767)
Unsigned16 ::= INTEGER(0..32767)
```

et en supposant que les valeurs à coder pour A et B soient 0x1234 et 0x5678[9] respectivement, le codage BER de cette séquence peut donner les séries d'octets suivantes:



Le codage A-XDR de la même séquence est le suivant:



9) Le terme 0x... indique que les chiffres suivants sont des chiffres hexadécimaux.

10) L'A-XDR requiert un identificateur de codage dans des cas particuliers seulement (par exemple, lorsque la SEQUENCE de cet exemple est l'un des choix d'un type CHOICE).

A-XDR encoding rules specify

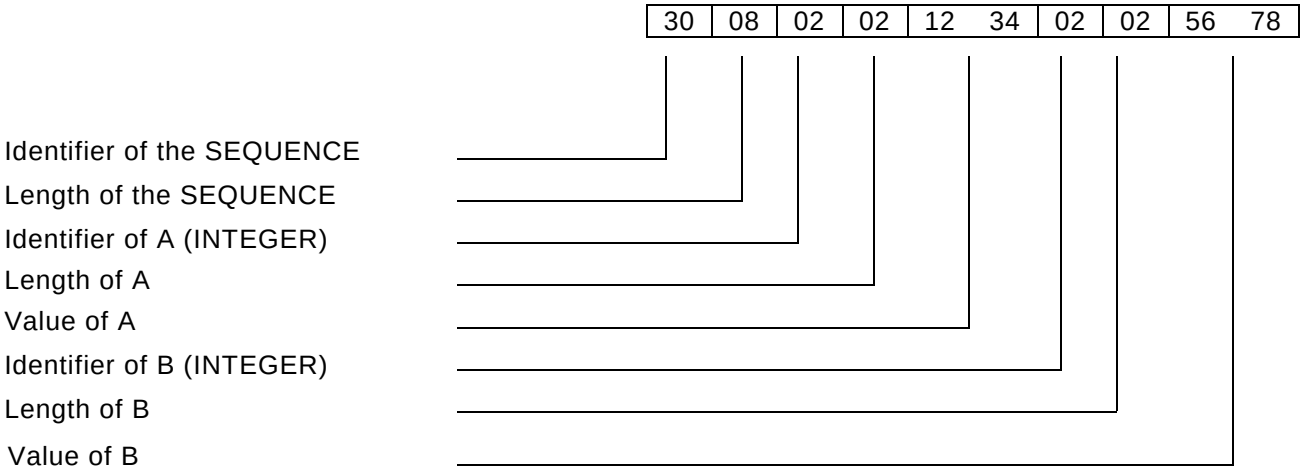
- encoding rules for the contents fields;
- the conditions when the L field should be present, and the encoding of that field;
- the conditions when the I field should be present, and the encoding of that field.

Example

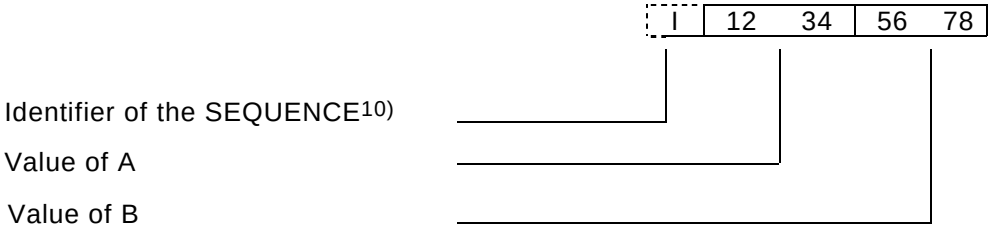
Taking the following ASN.1 composite type:

```
value ::= SEQUENCE {
        A           Integer16,
        B           Unsigned16
    }
Integer16 ::= INTEGER(–32768..32767)
Unsigned16 ::= INTEGER(0..32767)
```

and supposing that the values to be encoded for A and for B are 0x1234 and 0x5678[9] respectively, the BER encoding of that sequence may result in the following series of bytes:



A-XDR encoding of the same sequence is as follows:



[9] The term 0x... indicates that the following digits represent hexadecimal digits.

[10] A-XDR requires encoding identifier only in special cases (for example, when the SEQUENCE of this example is one of the choices of a CHOICE type).

5 Règles pour le codage

Le codage A-XDR de tout type ASN.1 donne un nombre entier d'octets contenant chacun huit bits. Cette série d'octets commence avec le premier octet du codage du champ Identificateur du type principal ASN.1 – ainsi, cet octet peut être considéré comme l'octet le plus significatif. Pour les besoins de la présente norme, le schéma d'identification suivant s'applique:

- les octets du codage A-XDR ne sont pas systématiquement numérotés mais parfois – lorsque ceci facilite la compréhension – des commentaires sont ajoutés (par exemple: 1er octet de la valeur, etc.);
- les bits de chaque octet sont numérotés de 1 à 8, le bit 8 étant le bit de poids fort.

5.1 Le champ Identificateur

L'objectif du champ Identificateur est d'indiquer le type de la valeur codée. Puisque l'émetteur et le récepteur utilisent exactement la même spécification de syntaxe abstraite, le champ Identificateur ne véhicule des informations que lorsque:

a) il convient de sélectionner un type de donnée parmi plusieurs alternatives – CHOICE;

b) il convient que la présence d'un composant OPTIONAL dans un type SEQUENCE soit indiquée;

c) il convient que la présence d'un composant DEFAULT dans un type SEQUENCE soit indiquée.

Le codage A-XDR ne contient un champ Identificateur que dans ces cas et lorsque la spécification ASN.1 l'exige (pour l'étiquetage explicite de l'ASN.1, voir 6.7).

Dans le cas a), A-XDR implique que toutes les alternatives du CHOICE soient spécifiées au niveau ASN.1 comme des types étiquetés explicitement (voir 6.7). Dans ces cas, l'étiquette codée constitue le champ Identificateur.

Dans les cas b) et c), la présence ou l'absence du composant OPTIONAL ou DEFAULT est indiquée par un «délimiteur de présence BOOLEAN». Le champ Identificateur pour ces valeurs de composant est le codage A-XDR de la valeur du délimiteur de présence (voir 6.9).

D'autre part, A-XDR peut être obligé de coder le champ Identificateur lorsque la définition ASN.1 contient des étiquettes explicites ASN.1 (voir 6.7). Le codage A-XDR de tels types est défini de façon à être identique à leur codage BER. Ce support a pour objectif de forcer le codage de la longueur, afin de permettre, par exemple, de contourner facilement certaines structures. Le champ Identificateur de ces types est la valeur de codage de l'étiquette ASN.1 et occupe un nombre entier d'octets, au minimum un octet, comme spécifié dans la Recommandation UIT-T X.209.

5 Rules for encoding

A-XDR encoding of any ASN.1 type results in an integral number of bytes, each containing eight bits. This series of bytes begins with the first byte of the encoding of the Identifier field of the outermost ASN.1 type – in these terms, this byte can be considered as the most significant byte. For the purpose of this standard, the following identification schema applies:

- the bytes of the A-XDR encoding are not systematically numbered, but sometimes, when it helps for clear understanding, comments apply (for example, 1 byte of the value, etc.);
- bits of any byte are numbered from 1 to 8, where bit 8 is the most significant.

5.1 The Identifier field

The purpose of the Identifier field is to indicate the type of the encoded value. Provided that the sender and the receiver are operating exactly the same specification of the abstract syntax, the Identifier field conveys information only in cases when

a) one type of data should be selected between different alternatives – CHOICE;

b) the presence of an OPTIONAL component in a SEQUENCE type should be indicated;

c) the presence of a DEFAULT component in a SEQUENCE type should be indicated.

A-XDR encoding contains an Identifier field only in these cases. In addition, A-XDR encodes an Identifier when encoding the Identifier is required by the ASN.1 specification (ASN.1 explicit tagging, see 6.7).

In case a), A-XDR requires that all the alternatives of the CHOICE will be specified at ASN.1 level as explicitly tagged types (see 6.7). In these cases, the encoded tag forms the identifier field.

In cases b) and c), the presence or the absence of the OPTIONAL or DEFAULT component is indicated by a so-called BOOLEAN presence flag. The Identifier field for these component values is the A-XDR encoding of the value of the presence flag (see 6.9).

On the other hand, A-XDR may be forced to encode the Identifier field, when the ASN.1 definition contains ASN.1 explicit tags (see 6.7). A-XDR encoding of such types is defined to be the same as their BER encoding. The aim of this support is to force the length to be encoded, for example, in order to allow for the easy omission of some structures. The Identifier field for those types is the encoded value of the ASN.1 tag and occupies an integral number of bytes, at least one, as specified in ITU-T Recommendation X.209.

5.2 Le champ Longueur

Dans A-XDR, le champ Longueur – lorsque ce champ est présent – précède immédiatement le champ Contenu, représente explicitement la longueur du champ Contenu et occupe un nombre entier d'octets. Puisque l'émetteur et le récepteur utilisent exactement la même spécification de syntaxe abstraite, le champ Longueur véhicule l'information seulement dans les cas où un type ASN.1 de longueur variable doit être codé. Les différents cas possibles sont:

a) INTEGER de longueur variable;

b) BIT STRING de longueur variable;

c) OCTET STRING de longueur variable;

d) Type SEQUENCE OF de longueur variable.

A-XDR code le champ Longueur seulement dans les cas mentionnés ci-avant ainsi que dans le cas où la spécification ASN.1 l'exige (étiquetage explicite ASN.1, voir 6.7).

Dans les cas a), b), c) et d), le champ Longueur est codé comme un nombre entier de longueur variable. Pour le cas particulier, le même codage s'applique, comme défini dans BER (voir Recommandation UIT-T X.209) à l'exception de la restriction selon laquelle seule la forme définie peut être utilisée. L'utilisation de la forme indéfinie (note de bas de page 7) pour le champ Longueur n'est pas autorisée pour l'A-XDR.

5.3 Le champ Contenu

Le champ Contenu est la substance du codage véhiculant la valeur réelle. Il se compose de zéro octet ou plus, et doit coder la valeur de donnée, comme spécifié dans les articles ci-après.

5.2 The Length field

In A-XDR the Length field (when this field is present) immediately precedes the Contents field, represents explicitly the length of the Contents field and occupies an integral number of bytes. Provided that the sender and the receiver are operating exactly the same specification of the abstract syntax, the Length field conveys information only in cases when a variable length ASN.1 type is to be encoded. The possible cases are

a) variable length INTEGER;

b) variable length BIT STRING;

c) variable length BYTE STRING;

d) variable length SEQUENCE OF type.

A-XDR encodes the Length field only in the above cases, and, in addition, in one more case, when it is required by the ASN.1 specification (ASN.1 explicit tagging, see 6.7).

In a), b), c) and d), the Length field is encoded as a variable length integer. For the additional case, the same encoding applies as defined in BER (see ITU-T Recommendation X.209) except for the restriction that only the definite form can be used. Using the indefinite form (footnote 7) for the Length field is not allowed within A-XDR.

5.3 The Contents field

The Contents field is the substance of the encoding, conveying the actual value. It consists of zero or more bytes, and shall encode the data value as specified in the following clauses.

6 Procédures de codage

6.1 Codage d'une valeur INTEGER

A-XDR fournit deux types de codage pour le type INTEGER ASN.1, selon que la définition ASN.1 d'un INTEGER est une valeur – contrainte ou non. Lorsqu'il est spécifié qu'un INTEGER est compris dans une plage réduite de valeurs, par exemple INTEGER(–128..127), il est codé comme un *nombre entier de longueur fixe*. Sinon, lorsque aucune plage n'est spécifiée, l'INTEGER est codé comme un *nombre entier de longueur variable*.

6.1.1 Codage d'une valeur de nombre entier de longueur fixe

Deux codages différents sont fournis par A-XDR pour des nombres entiers de longueur fixe: les nombres entiers spécifiés dans une plage de valeur non négative sont représentés et codés comme des nombres *binaires sans signe* alors que les nombres entiers susceptibles de prendre de valeurs négatives sont représentés et codés comme deux nombres binaires complémentaires. Dans les deux cas, seule la valeur du nombre entier est codée, formant le champ de contenu du codage. L'objectif est de fournir un codage de longueur minimale.

6.1.1.1 Codage de valeurs de nombre entier sans signe, de longueur fixe

Lorsqu'il est spécifié qu'un INTEGER est compris dans une plage de valeur non négative, il est codé comme un nombre binaire sans signe. Le nombre d'octets du codage est déterminé par la plage de valeur spécifiée: il est égal au nombre minimal d'octets nécessaires requis pour représenter toute valeur comprise dans la plage spécifiée. La plage d'un nombre entier sans signe de longueur fixe est toujours alignée sur la limite d'octets.

Exemple

INTEGER(0..255) est codé en 1 octet

INTEGER(0..256) est codé en 2 octets

INTEGER(237..256) est codé en 2 octets

Un nombre entier sans signe de longueur fixe est présenté en notation binaire sans signe. Le codage est un nombre binaire sans signe égal à la valeur du nombre entier et se compose des bits 8 à 1 du premier octet, suivis des bits 8 à 1 du second octet, suivis par les bits 8 à 1 de chaque octet l'un après l'autre, y compris le dernier octet du codage.

Exemple

Un codage A-XDR de la valeur 61478 d'un INTEGER(0..65535) est le suivant:

1er octet	2e octet
87654321	87654321
11110000	**00100110**

6 Encoding procedures

6.1 Encoding of an INTEGER value

A-XDR provides two types of encoding for the ASN.1 INTEGER type, depending on whether the ASN.1 definition of an INTEGER is value-constrained or not. When an INTEGER is specified to be within a restricted range of values, for example INTEGER(–128..127), it is encoded as a *fixed-length integer*. Otherwise, when no range is specified, the INTEGER is encoded as a *variable-length integer*.

6.1.1 Encoding of a fixed-length integer value

Two different encodings are provided by A-XDR for fixed-length integers: integers which are specified to be within a non-negative value range are represented and encoded as *unsigned* binary numbers, while integers which may take negative values are represented and encoded as two's complement binary numbers. In both cases, only the value of the integer is encoded, forming the contents field of the encoding. The aim is to provide minimal-length encoding.

6.1.1.1 Encoding of fixed-length, unsigned integer values

When an INTEGER is specified to be within a non-negative value range, it is encoded as an unsigned binary number. The number of bytes of the encoding is determined by the specified value-range: it is equal to the minimum number of necessary bytes which is required to represent any value within the specified range. The range of a fixed-length unsigned integer is always aligned on byte boundary.

Example

INTEGER(0..255) is encoded in 1 byte

INTEGER(0..256) is encoded in 2 bytes

INTEGER(237..256) is encoded in 2 bytes

A fixed-length unsigned integer is presented in unsigned binary notation. The encoding is an unsigned binary number equal to the integer value and consists of bits 8 to 1 of the first byte, followed by bits 8 to 1 of the second byte, followed by bits 8 to 1 of each byte in turn, up to and including the last byte of the encoding.

Example

A-XDR encoding of the 61478 value of an INTEGER(0..65535) is as follows:

1st byte	2nd byte
8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
1 1 1 1 0 0 0 0	**0 0 1 0 0 1 1 0**

6.1.1.2 Codage de valeurs de nombre entier avec signe de longueur fixe

Lorsqu'un INTEGER est spécifié comme étant compris dans une plage de valeurs incluant des valeurs négatives, il est codé comme un nombre binaire complément à deux. Le nombre d'octets du codage est déterminé par la plage de valeurs spécifiée: il est égal au nombre minimal d'octets nécessaires requis pour représenter toute valeur comprise dans la plage spécifiée. La plage d'un nombre entier de longueur fixe est toujours alignée sur la limite d'octets.

Exemple

INTEGER(–32768..32767) est codé en 2 octets,

INTEGER(–14300..8700) est également codée en 2 octets, et

INTEGER(–32768..32768) est codée en 3 octets

Le codage est un nombre binaire complément à deux égal à la valeur du nombre entier et se compose des bits 8 à 1 du premier octet, suivis des bits 8 à 1 du second octet, suivis par les bits 8 à 1 de chaque octet l'un après l'autre, y compris le dernier octet du codage.

Exemple

Le codage A-XDR de la valeur –45783 d'un INTEGER(–50000..1) est le suivant:

1er octet	2e octet	3e octet
8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
1 1 1 1 1 1 1 1	**0 1 0 0 1 1 0 1**	**0 0 1 0 1 0 0 1**

6.1.2 Codage d'une valeur de nombre entier de longueur variable

Lorsque la plage de valeurs d'un INTEGER n'est pas spécifiée, celui-ci est codé comme un nombre entier de longueur variable. Le codage d'un nombre entier de longueur variable peut être donné sous deux formes, selon la valeur à coder.

Lorsque la valeur d'un INTEGER non contrainte est comprise entre 0 et 127, (0 ≤ valeur < 128), seule la valeur est codée comme un octet individuel. Le bit 8 de cet octet est évidemment 0 (zéro).

Exemple

Le codage A-XDR de la valeur d'INTEGER 123 (= 0x7B) est le suivant:

8 7 6 5 4 3 2 1
0 1 1 1 1 0 1 1

6.1.1.2 Encoding of fixed-length, signed integer values

When an INTEGER is specified to be within a value range which includes negative values, it is encoded as a two's complement binary number. The number of bytes of the encoding is determined by the specified value-range; it is equal to the minimum number of necessary bytes which is required to represent any value within the specified range. The range of a fixed-length integer is always aligned on byte boundary.

Example

INTEGER(–32768..32767) is encoded in 2 bytes

INTEGER(–14300..8700) is also encoded in 2 bytes

INTEGER(–32768..32768) is encoded in 3 bytes

The encoding is a two's complement binary number equal to the integer value, and consists of bits 8 to 1 of the first byte, followed by bits 8 to 1 of the second byte, followed by bits 8 to 1 of each byte in turn, up to and including the last byte of the encoding.

Example

A-XDR encoding of the –45783 value of an INTEGER(–50000..1) is as follows:

1st byte	2nd byte	3rd byte
8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
1 1 1 1 1 1 1 1	**0 1 0 0 1 1 0 1**	**0 0 1 0 1 0 0 1**

6.1.2 Encoding of a variable-length integer value

When the range of values of an INTEGER is not specified, that INTEGER is encoded as a variable-length integer. Encoding of a variable-length integer may be given in two forms, depending on the value to be encoded.

When the value of a non-constrained INTEGER is between 0 and 127 ($0 \leq$ value < 128), only the value is encoded as a single byte. Bit 8 of this byte is obviously 0 (zero).

Example

A-XDR encoding of the INTEGER value 123 (= 0x7B) is as follows:

8 7 6 5 4 3 2 1
0 1 1 1 1 0 1 1

Le codage d'une valeur d'INTEGER non contrainte, lorsqu'elle n'est pas comprise dans la plage 0 ≤ valeur < 128 mentionnée ci-dessus, comprend deux champs: le premier champ, de longueur fixe – Longueur –, représente la longueur du second champ de longueur variable, en octets. Ce second champ de longueur variable – Contenu – véhicule la valeur codée et contient un nombre entier d'octets.

Le champ Longueur est codé en un octet[11)], avec le bit 8 réglé sur 1 (indiquant la présence du champ Longueur). Les sept bits restants sont codés comme un nombre entier sans signe de longueur fixe, dont la valeur représente le nombre d'octets dans le champ de contenu.

Le nombre d'octets du codage est déterminé par la valeur à coder: il est égal au nombre minimal d'octets nécessaires requis pour représenter la valeur donnée.

Le codage de la valeur est un nombre binaire complémentaire double égal à la valeur du nombre entier et se compose des bits 8 à 1 du premier octet, suivi par les bits 8 à 1 du second octet, suivi par les bits 8 à 1 de chaque octet l'un après l'autre, y compris le dernier octet du codage – comme il est également spécifié pour les nombres entiers signés de longueur fixe (6.1.1.2). La structure du codage d'un nombre entier de longueur variable est décrite à la figure 4.

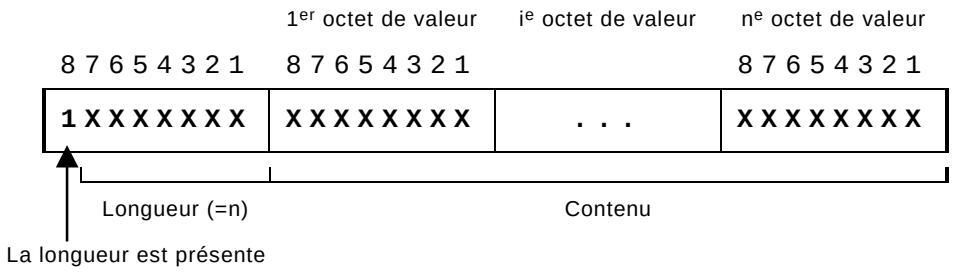
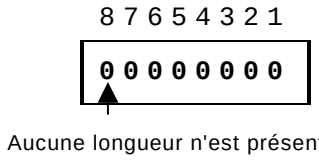


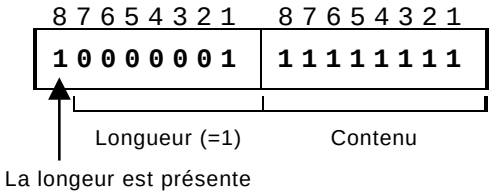
Figure 4 – Structure du codage d'un nombre entier de longueur variable

Exemples de codage A-XDR de nombres entiers de longueur variable

a) Le codage de la valeur 0 (zéro) est le suivant:



b) Le codage de la valeur –1 est le suivant:



11) La valeur du champ Longueur peut correspondre à toute valeur comprise entre 0...127, ce qui permet le codage des INTEGERS représentés par 1016 bits maximum.

Encoding of a non-constrained INTEGER value when it is out of the above 0 ≤ value < 128 range contains two fields: the first, fixed-length field – Length – represents the length of the second, variable-length field, in bytes. This second, variable-length field – Contents – conveys the encoded value, and contains an integral number of bytes.

The Length field is encoded in one byte[11], with bit 8 set to 1 (indicating the presence of the Length field). The remaining seven bits are encoded as a fixed-length, unsigned integer, the value of which represents the number of bytes in the contents field.

The number of bytes of the encoding is determined by the value to be encoded: it is equal to the minimum number of necessary bytes which is required to represent the given value.

The encoding of the value is a two's complement binary number equal to the integer value, and consists of bits 8 to 1 of the first byte, followed by bits 8 to 1 of the second byte, followed by bits 8 to 1 of each byte in turn, up to and including the last byte of the encoding – as specified also for fixed-length, signed integers (6.1.1.2). The structure of a variable length integer encoding is shown in figure 4.

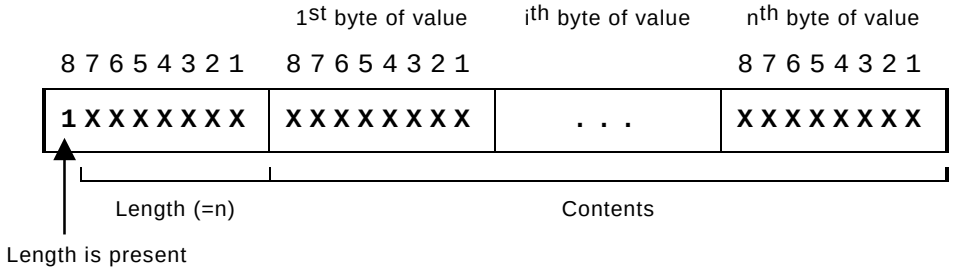
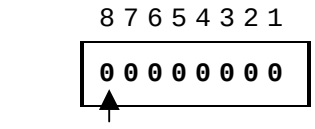


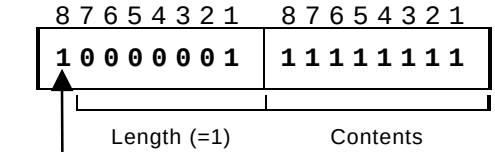
Figure 4 – Structure of the variable-length integer encoding

Examples of A-XDR encoding of variable-length integers

a) Encoding of the 0 (zero) value is as follows:

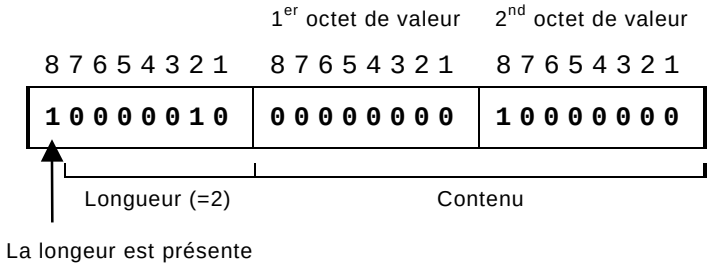


b) Encoding of the –1 value is as follows:

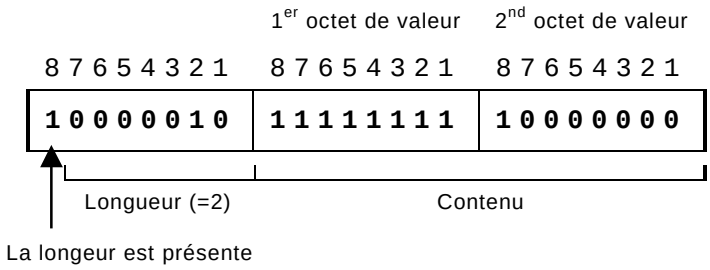


[11] The value of the Length field thus can be any value between 0..127, which allows the encoding of INTEGERS represented in maximum 1016 bits.

c) Le codage de la valeur 128 est le suivant:



d) Le codage de la valeur –128 est le suivant:

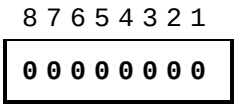


6.2 Codage de la valeur BOOLEAN

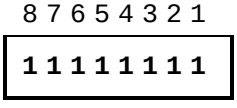
Un type BOOLEAN peut prendre seulement deux valeurs: il est TRUE ou FALSE. Un codage A-XDR d'une valeur BOOLEAN comporte seulement le champ Contenu se composant d'un octet. Si la valeur à coder est FALSE, cet octet est zéro (tous les bits sont à zéro), dans le cas contraire – lorsque la valeur est TRUE – cet octet peut prendre n'importe quelle valeur différente de zéro, comme une option de l'émetteur.

Exemple

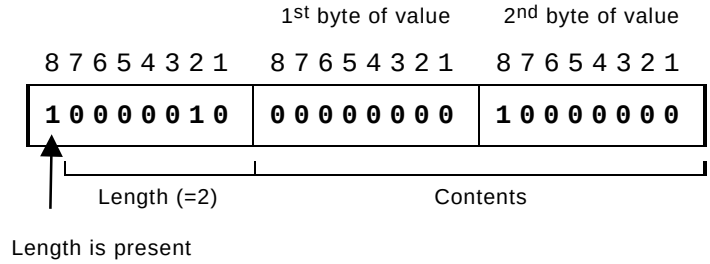
Le codage A-XDR de la valeur FALSE BOOLEAN est le suivant:



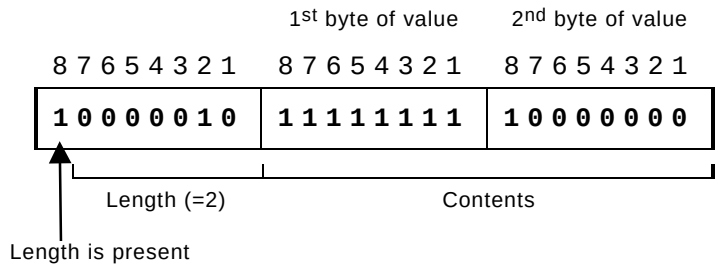
Un codage valide de la valeur TRUE BOOLEAN est le suivant:



c) Encoding of the 128 value is as follows:



d) Encoding of the –128 value is as follows:

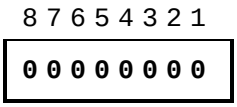


6.2 Encoding of a BOOLEAN value

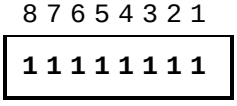
A BOOLEAN type may take only two values: it is either TRUE or FALSE. A-XDR encoding of a BOOLEAN value contains only the Contents field, which consists of one byte. If the value to be encoded is FALSE, this byte is zero (all bits are zero), otherwise – when the value is TRUE – this byte can be any non-zero value, as a sender's option.

Example

A-XDR encoding of the FALSE BOOLEAN value is as follows:



A valid encoding of the TRUE BOOLEAN value is as follows:



6.3 Codage d'une valeur ENUMERATED

La plage de valeurs pour le type ENUMERATED est réduite entre 0..255 pour le codage A-XDR. Le codage A-XDR d'une valeur ENUMERATED doit être celui de la valeur de l'INTEGER contrainte (0..255) codée comme nombre entier sans signe de longueur fixe en un octet.

6.4 Codage d'une valeur BIT STRING

A-XDR fournit deux types de codage pour le type BIT STRING ASN.1, selon si la définition ASN.1 du BIT STRING spécifie la taille du BIT STRING ou non. Un codage *en longueur fixe* est assuré lorsque la taille d'un BIT STRING est spécifiée dans la définition de l'ASN.1 alors que les BIT STRINGS dont la taille n'est pas spécifiée sont codés en *longueur variable*. Dans les deux cas, le codage d'une valeur de BIT STRING est aligné sur la limite d'octet en ajoutant des bits finaux de valeur 0.

6.4.1 Codage en longueur fixe pour les valeurs BIT STRING dont la taille est spécifiée

Si la taille du BIT STRING est spécifiée dans la description ASN.1, le codage A-XDR de ce BIT STRING comporte seulement un champ Contenu. Le nombre d'octets dans le champ de Contenu est déterminé par la taille spécifiée: il est égal au nombre minimal d'octets nécessaires pour véhiculer le nombre de bits dans le BIT STRING.

Exemple

BIT STRING(SIZE(3)) est codé avec 1 octet,

BIT STRING(SIZE(8)) est codé avec 1 octet,

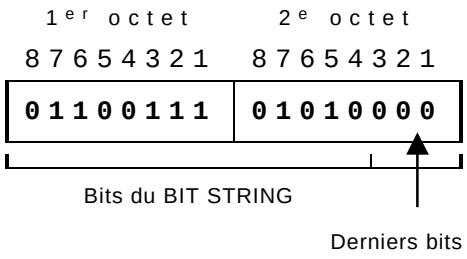
BIT STRING(SIZE(14)) est codé avec 2 octets.

Les bits du BIT STRING commençant au premier bit et continuant jusqu'au(x) dernier(s) bit(s) doivent être placés en bits 8 à 1 du premier octet suivant, suivi par les bits 8 à 1 du second octet suivant, suivi par les bits 8 à 1 de chaque octet l'un après l'autre, suivi par autant de bits que nécessaire de l'octet suivant final, en commençant par le bit 8.

Chaque octet du codage, à l'exception du dernier, doit comporter huit bits de la BIT STRING. Le dernier octet du codage doit comporter les bits restants de la BIT STRING en ajoutant des bits finaux de valeur 0.

Exemple

Le codage A-XDR de la valeur 01100111 01010 de la BIT STRING(SIZE(13)) est le suivant:



6.3 Encoding of an ENUMERATED value

The value range for the ENUMERATED type is restrained for A-XDR encoding between 0..255. A-XDR encoding of an ENUMERATED value shall be that of the constrained INTEGER(0..255) value encoded as a fixed-length, unsigned integer in one byte.

6.4 Encoding of a BIT STRING value

A-XDR provides two types of encoding for the ASN.1 BIT STRING type, depending on whether the ASN.1 definition of the BIT STRING specifies the size of the BIT STRING or not. *Fixed-length* encoding is provided when the size of a BIT STRING is specified in the ASN.1 definition, while BIT STRINGS with no specified size are encoded in *variable-length* manner. In both cases, the encoding of a BIT STRING value is aligned on byte boundary by adding 0 value trailing bits.

6.4.1 Fixed-length encoding for specified size BIT STRING values

If the size of the BIT STRING is specified in the ASN.1 description, the A-XDR encoding of that BIT STRING contains only a Contents field. The number of bytes in the Contents field is determined by the specified size: it is equal to the minimum number of necessary bytes to convey the number of bits in the BIT STRING.

Example

BIT STRING(SIZE(3)) is encoded in 1 byte

BIT STRING(SIZE(8)) is encoded in 1 byte

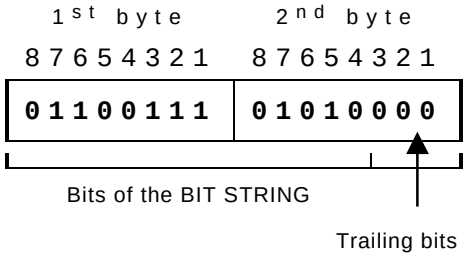
BIT STRING(SIZE(14)) is encoded in 2 bytes

The bits in the BIT STRING, commencing with the first bit and proceeding to the trailing bit(s), shall be placed in bits 8 to 1 of the first subsequent byte, followed by bits 8 to 1 of the second subsequent byte, followed by bits 8 to 1 of each byte in turn, followed by as many bits as are needed of the final subsequent byte, commencing with bit 8.

Each byte of the encoding, with the exception of the last, shall contain eight bits of the BIT STRING. The last byte of the encoding shall contain the remaining bits of the BIT STRING followed by 0 value trailing bits.

Example

A-XDR encoding of the 01100111 01010 value of BIT STRING(SIZE(13)) is as follows:



6.4.2 Codage en longueur variable pour les valeurs de BIT STRING dont la taille n'est pas spécifiée

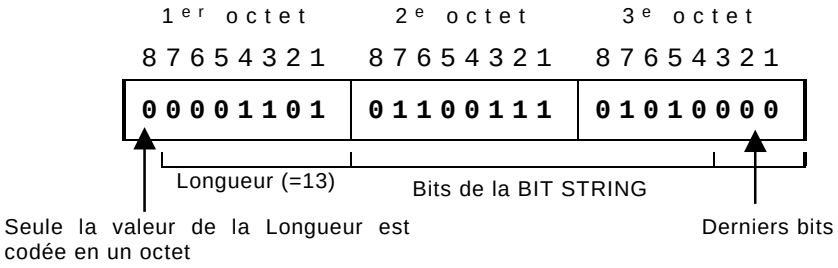
Le codage A-XDR d'une valeur de BIT STRING, lorsque la taille du BIT STRING n'est pas spécifiée dans la définition ASN.1, comporte deux champs: un champ Longueur et un champ Contenu.

La valeur représentée par le champ Longueur est égale au nombre de bits de la valeur codée du BIT STRING. Le champ Longueur lui-même est codé de façon similaire au codage du nombre entier de longueur variable, sauf que les nombres entiers sont codés comme des nombres binaires et non comme des nombres binaires compléments à deux, car les valeurs négatives n'ont aucune signification pour le champ Longueur. (Cette opération est similaire au codage pour les nombres entiers sans signe de longueur variable.)

Le champ Contenu véhicule la valeur codée du BIT STRING et comporte un nombre entier d'octets. Les règles de codage pour ce champ sont les mêmes que celles spécifiées pour les valeurs de BIT STRING de longueur fixe au 6.4.1.

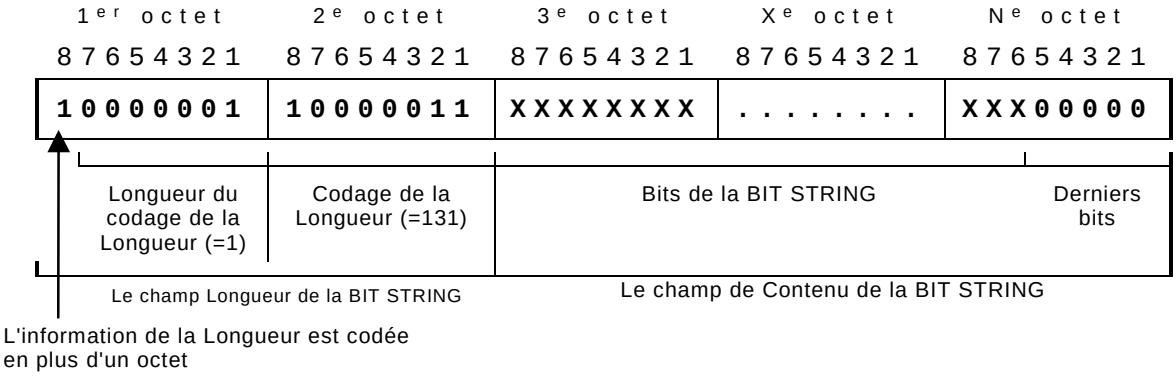
Exemple

Le codage A-XDR de la valeur 01100111 01010 d'un BIT STRING est le suivant:



Exemple

Le codage A-XDR d'une valeur de BIT STRING composé de 131 bits est le suivant:



6.5 Codage d'une valeur OCTET STRING

A-XDR fournit deux types de codage pour le type OCTET STRING ASN.1, selon que la définition ASN.1 de l'OCTET STRING spécifie la taille de l'OCTET STRING ou non. Un codage *de longueur fixe* est effectué lorsque la taille de l'OCTET STRING est spécifiée dans la définition ASN.1 alors que les OCTET STRINGS dont la taille n'est pas spécifiée sont codés en *longueur variable*.

6.4.2 Variable-length encoding for non-specified size BIT STRING values

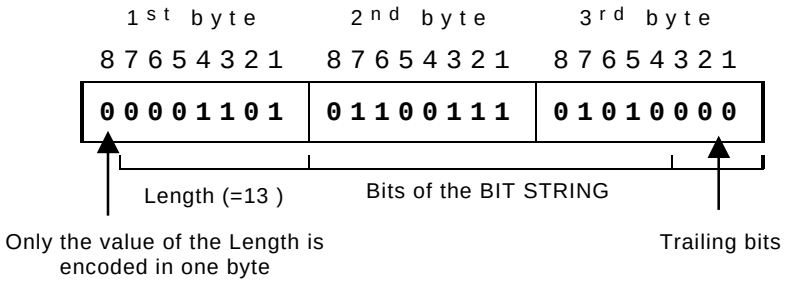
A-XDR encoding of a BIT STRING value when the size of the BIT STRING is not specified in the ASN.1 definition contains two fields: a Length field and a Contents field.

The value represented by the Length field is equal to the number of bits in the encoded value of the BIT STRING. The Length field itself is encoded similarly to the variable-length integer encoding, except that (because negative values have no meaning for the Length field) integers are encoded as binary numbers and not as two's complement binary numbers. (It is like an encoding for variable-length unsigned integers.)

The Contents field conveys the encoded value of the BIT STRING, and contains an integral number of bytes. Encoding rules for this field are the same as specified for the fixed-length BIT STRING values in 6.4.1.

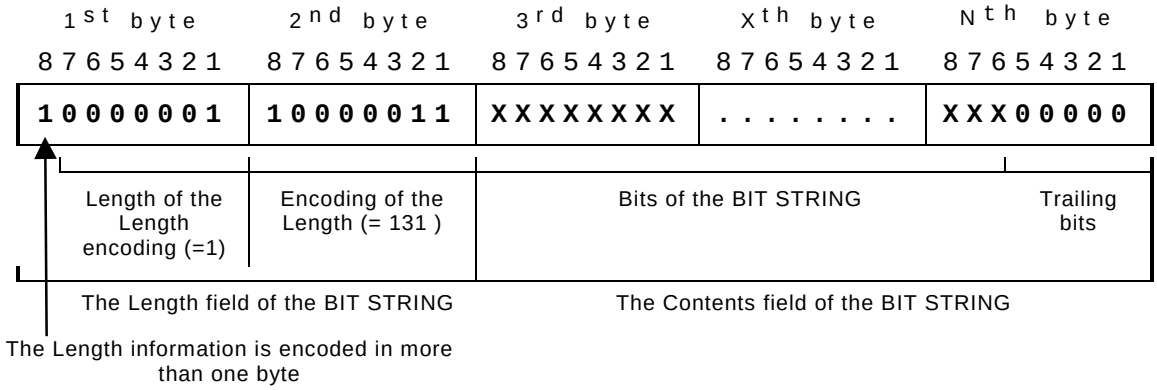
Example

A-XDR encoding of the 01100111 01010 value of a BIT STRING is as follows:



Example

A-XDR encoding of a BIT STRING value consisting of 131 bits is as follows:



6.5 Encoding of an BYTE STRING value

A-XDR provides two types of encoding for the ASN.1 BYTE STRING type, depending on whether the ASN.1 definition of the BYTE STRING specifies the size of the BYTE STRING or not. *Fixed-length* encoding is provided when the size of the BYTE STRING is specified in the ASN.1 definition, while BYTE STRINGS with no specified size are encoded in *variable-length* manner.

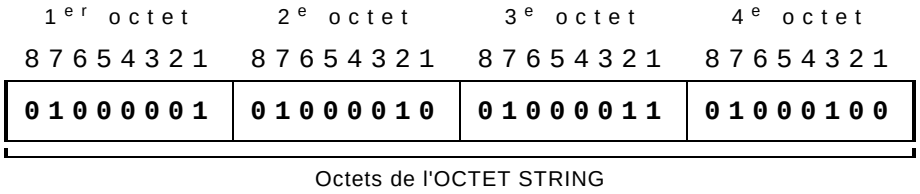
6.5.1 Codage en longueur fixe pour les valeurs OCTET STRING dont la taille est spécifiée

Si la taille de l'OCTET STRING est spécifiée dans la description ASN.1, le codage A-XDR de cette BIT STRING contient seulement un champ Contenu. Le nombre d'octets du champ Contenu est égal à la taille spécifiée.

Les octets de l'OCTET STRING, commençant avec le premier octet et continuant jusqu'au dernier octet doivent simplement être placés dans les octets du champ Contenu.

Exemple

Le codage A-XDR de la valeur de OCTET STRING(SIZE(4)) «ABCD» est le suivant:



6.5.2 Codage en longueur variable pour les valeurs d'OCTET STRING dont la taille n'est pas spécifiée

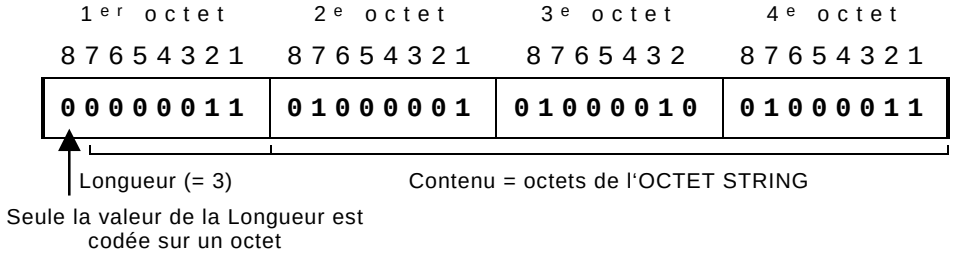
Le codage A-XDR d'une valeur OCTET STRING lorsque la taille OCTET STRING n'est pas spécifiée dans la définition ASN.1 comporte deux champs: un champ Longueur et un champ Contenu.

La valeur représentée par le champ Longueur est égale au nombre d'octets dans le champ Contenu. Le champ Longueur lui-même est codé exactement de la même manière que celle spécifiée pour le codage du champ Longueur des BIT STRINGS de longueur variable (voir 6.4.2).

Le champ Contenu véhicule la valeur codée OCTET STRING. Les règles de codage pour ce champ sont les mêmes que celles spécifiées pour les valeurs OCTET STRING de longueur fixe au 6.5.1.

Exemple

Le codage A-XDR de la valeur «ABC» d'un OCTET STRING est le suivant:



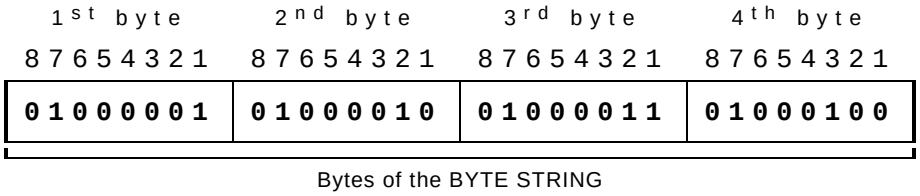
6.5.1 Fixed-length encoding for specified size BYTE STRING values

If the size of the BYTE STRING is specified in the ASN.1 description, the A-XDR encoding of that BIT STRING contains only a Contents field. The number of bytes in the Contents field is equal to the specified size.

The bytes in the BYTE STRING, commencing with the first byte and proceeding to the last byte shall simply be placed in the bytes of the Contents field.

Example

A-XDR encoding of the value of "ABCD" BYTE STRING(SIZE(4)) is as follows:



6.5.2 Variable-length encoding for non-specified size BYTE STRING values

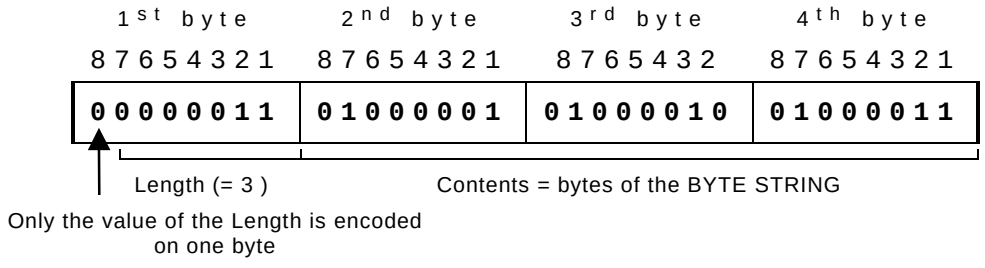
A-XDR encoding of an BYTE STRING value when the size of the BYTE STRING is not specified in the ASN.1 definition contains two fields: a Length field and a Contents field.

The value represented by the Length field is equal to the number of bytes in the Contents field. The Length field itself is encoded exactly in the same way as specified for the Length field encoding of variable-length BIT STRINGS (see 6.4.2).

The Contents field conveys the encoded value of the BYTE STRING. Encoding rules for this field are the same as specified for the fixed-length BYTE STRING values in 6.5.1.

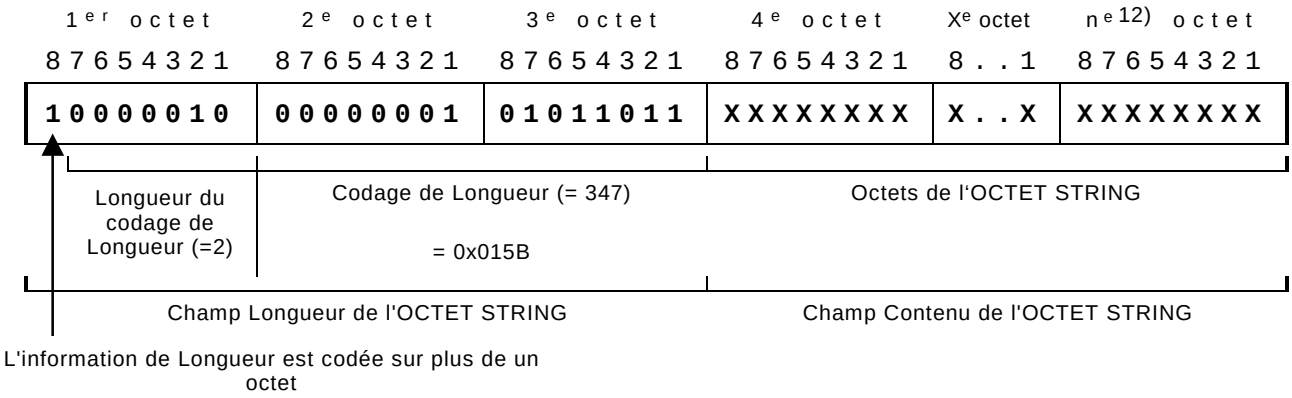
Example

A-XDR encoding of the "ABC" value of an BYTE STRING is as follows:



Exemple

Le codage A-XDR d'une valeur d'OCTET STRING se compose de 347 octets:



6.6 Codage de la valeur CHOICE

L'un des principaux concepts d'A-XDR est que lorsque l'émetteur et le récepteur d'un message utilisent tous les deux la même syntaxe abstraite, dans la plupart des cas, le champ identificateur d'un codage BER véhicule une information redondante. Dans ces cas, le fait de ne pas coder le champ Identificateur n'entraîne pas de codage ambigu. Par conséquent, le codage A-XDR ne code pas systématiquement l'identificateur pour les types ASN.1.

Le type CHOICE ASN.1 est défini comme un groupe de types de composants, ses alternatives, qui doivent être distincts les uns des autres. Chaque valeur d'un type CHOICE est seulement une valeur de l'une des alternatives[13]. Il convient que les règles de codage garantissent que la valeur codée de chaque alternative soit identifiée sans ambiguïté – ainsi, le codage A-XDR d'une valeur d'un type CHOICE doit contenir un champ d'identification.

Afin d'éliminer toute ambiguïté, il convient que tous les types de composants de la spécification ASN.1 d'un type CHOICE soient un type explicitement étiqueté[14]. Les types CHOICE avec des composants non explicitement étiquetés ne peuvent pas être codés en A-XDR.

Un codage A-XDR d'une valeur CHOICE doit être le codage A-XDR d'une valeur du type alternatif choisi précédé par l'identificateur (étiquette), codé en un octet comme un nombre entier sans signe de longueur fixe.

Exemple

Considérons la structure CHOICE ASN.1 suivante:

```
Dummy_PDU ::= CHOICE {
        a       [0] INTEGER
        b       [1] OCTET STRING(SIZE(4))
}
```

_____

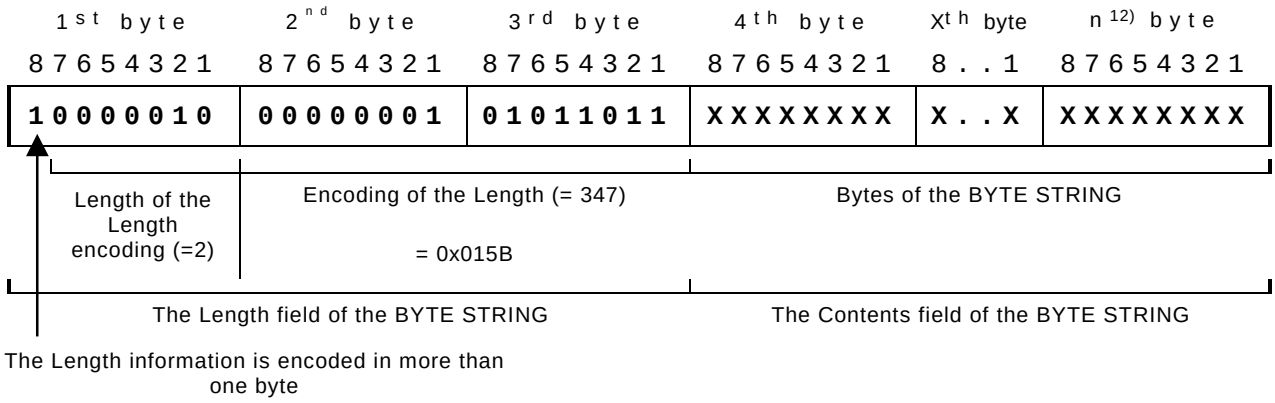
[12] n = taille de (champ Longueur en octets) + taille de (champ Contenu en octets) = 3 + 347 = 350

[13] Le type CHOICE ASN.1 est similaire au type UNION C.

[14] Le terme «étiquette» sera défini en 6.7.

Example

A-XDR encoding of an BYTE STRING value consisting of 347 bytes:



6.6 Encoding of a CHOICE value

One of the principal concepts of A-XDR is that, when both the sender and the receiver of a message operate the same abstract syntax, in most cases the Identifier field of a BER encoding conveys redundant information. In these cases, not encoding the Identifier field does not result in unambiguous encoding. Consequently, A-XDR encoding does not systematically encode the identifier for ASN.1 types.

The CHOICE ASN.1 type is defined in terms of a collection of component types, its alternatives, which must all be distinct. Each value of a CHOICE type is a value of just one of the alternatives[13]. The encoding rules should ensure that the encoded value of any of the alternatives might be unambiguously identified – therefore A-XDR encoding of a value of a CHOICE type shall contain an identifier field.

To ensure unambiguity, all component types of the ASN.1 specification of a CHOICE type should be an explicitly tagged[14] type. CHOICE types with not explicitly tagged components cannot be encoded in A-XDR.

A-XDR encoding of a CHOICE value shall be the A-XDR encoding of a value of the chosen alternative type preceded by the identifier (tag), encoded in one byte as a fixed-length unsigned integer.

Example
Taking the following ASN.1 CHOICE structure:

```
Dummy_PDU::= CHOICE {
    a           [0] INTEGER,
    b           [1] BYTE STRING(SIZE(4))
}
```

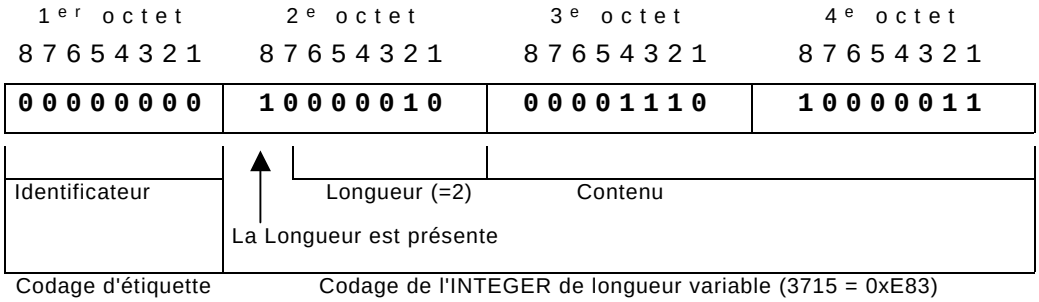
---

12) n = size of (Length field in bytes) + size of (Contents field in bytes) = 3 + 347 = 350

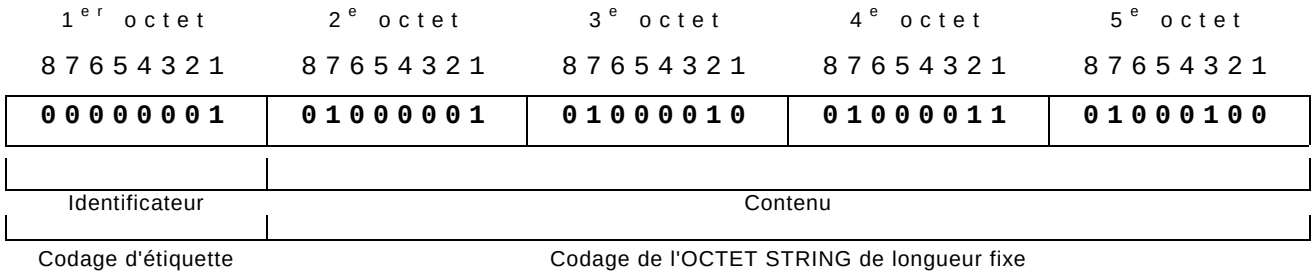
13) The ASN.1 CHOICE type is similar to the C UNION type.

14) The term 'tag' will be defined in 6.7.

Le codage A-XDR de la valeur de ce type CHOICE lorsque l'alternative 'a' est choisie et que a = 3715 est le suivant:



Lorsque l'alternative 'b' est choisie, pour une valeur de b = «ABCD», le codage A-XDR de la valeur type CHOICE est la suivante:



6.7 Types étiquetés (étiquetage implicite, explicite et explicite ASN.1)

Reprenons la Dummy_PDU spécifiée en 6.6:

```
Dummy_PDU ::= CHOICE {
    a       [0] INTEGER
    b       [1] OCTET STRING(SIZE(4))
}
```

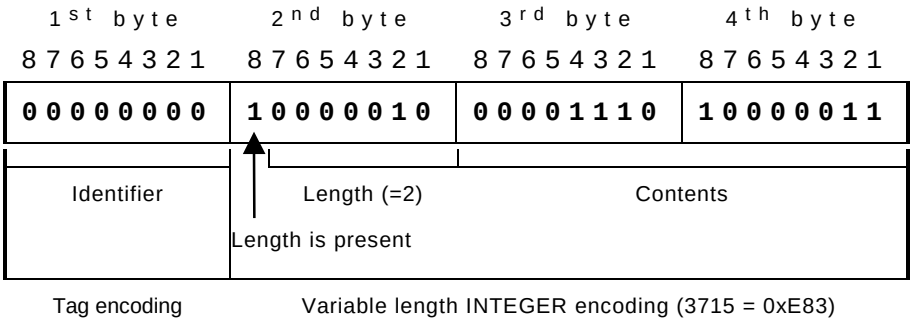
Les notations [0] INTEGER et [1] OCTET STRING (obtenues en faisant précéder la notation originale du type par un numéro entre crochets) représentent la notation type ASN.1 pour les *types étiquetés*. L'étiquetage permet de garantir une distinction.

En fait, chaque type ASN.1 (y compris les types ASN.1 intégrés comme INTEGER, BIT STRING, etc.) ont une étiquette[15]. Quatre classes d'étiquettes distinctes sont spécifiées et chaque étiquette est numérotée dans sa classe avec un nombre entier positif. Les quatre classes sont: universelle, spécifique au contexte, étendue à l'application et usage privé. Les types ASN.1 intégrés ont des étiquettes de la classe universelle et sont nommés types de base. Les étiquettes de toutes les classes autres que la classe universelle permettent de former les types étiquetés.

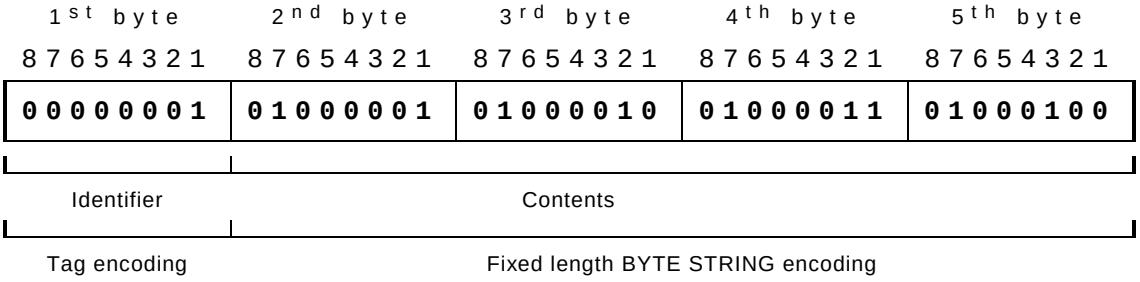
Chaque étiquette est implicite ou explicite. La sélection peut être effectuée dans la notation du type étiqueté en ajoutant le mot-clef IMPLICIT ou EXPLICIT entre le "]" et le type de base. Si aucun mot-clef n'apparaît, le style d'étiquette par défaut s'applique. Noter que les mots-clefs IMPLICIT et EXPLICIT *qualifient l'étiquette* (et non le type) et concernent seulement le codage.

15) A l'exception des types CHOICE qui ont une étiquette pour chacune de leurs alternatives et du type ANY qui a toutes les étiquettes possibles.

A-XDR encoding of the value of that CHOICE type when the ‘a’ alternative is chosen, and a = 3715 is as follows:



When the ‘b’ alternative is chosen, with the value of b = "ABCD", A-XDR encoding of the value of the CHOICE type is as follows:



6.7 Tagged types (implicit, explicit and ASN.1 explicit tagging)

Coming back to the Dummy_PDU, specified in 6.6:

```
Dummy_PDU::=CHOICE {
        a       [0] INTEGER,
        b       [1] BYTE STRING(SIZE(4))
}
```

The notations [0] INTEGER and [1] BYTE STRING (obtained by preceding the original type notation by a number in square brackets) is the ASN.1 type notation for *tagged types*. Tagging serves to ensure distinctness.

In fact, every ASN.1 type (even the built-in ASN.1 types like INTEGER, BIT STRING, etc.) has a tag[15]. Four distinct tag classes are specified, and each tag within its class is numbered with a non-negative integer. The four classes are: universal, context-specific, application-wide and private-use. Built-in ASN.1 types have tags in the universal class and are called base types. The tags of all classes other than universal serve to form tagged types.

Every tag is either implicit or explicit. The selection can be made in the tagged type notation by adding the keyword IMPLICIT or EXPLICIT between the "]" and the base type. If neither keyword appears, then the default tag style applies. Note that IMPLICIT and EXPLICIT keywords *qualify the tag* (and not the type) and affect only the encoding.

---

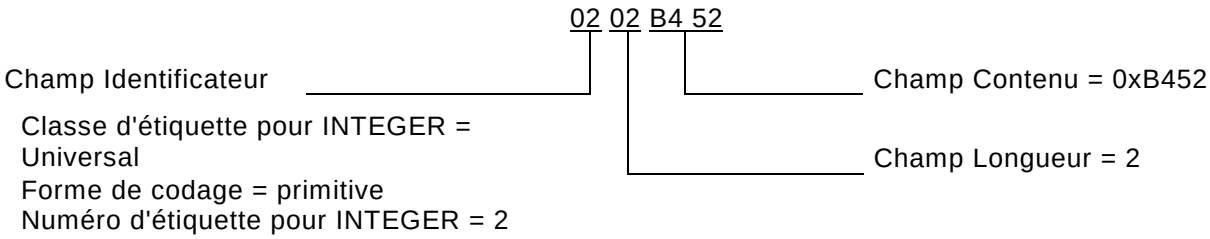
15) Except for CHOICE types, which have a tag for each of their alternatives and the ANY type which has all possible tags.

Pour l'encodage BER, la classe et le numéro de l'étiquette sont véhiculés par le champ Identificateur. Lorsque l'étiquetage est EXPLICIT, le codage BER de cette valeur réalise deux étiquettes: la nouvelle, spécifiée entre crochets ainsi que l'étiquette du type original, de base. Le codage est toujours construit, avec exactement un codage emboîté, le codage du type de base.

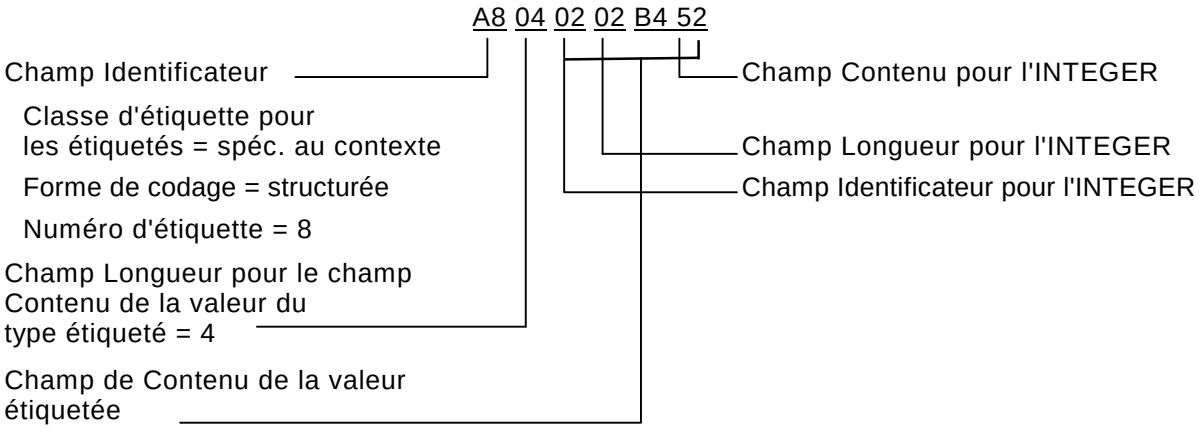
Lorsque l'étiquetage IMPLICIT est spécifié, le codage BER de la valeur étiquetée comporte seulement la nouvelle étiquette: elle remplace l'étiquette (classe et numéro) du type original.

Exemples

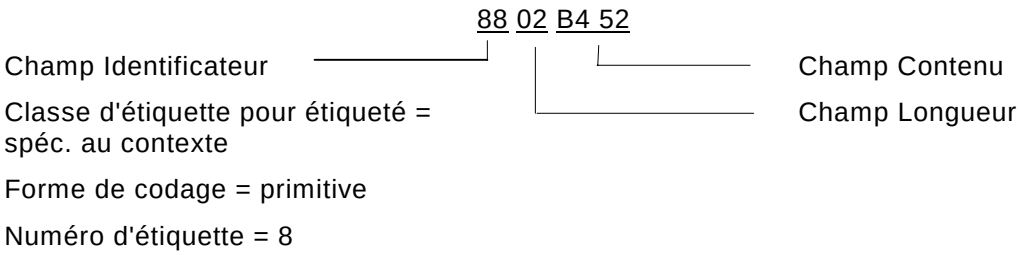
- Le codage BER de la valeur –19374 (0xB542) d'un type INTEGER est le suivant:



- Le codage BER de la même valeur mais étiquetée comme [8] INTEGER (explicitement étiquetée: il s'agit du style d'étiquette par défaut du BER) est le suivant:



- Le codage BER de la même valeur mais étiquetée comme [8] INTEGER IMPLICITE (implicitement étiqueté) est le suivant:

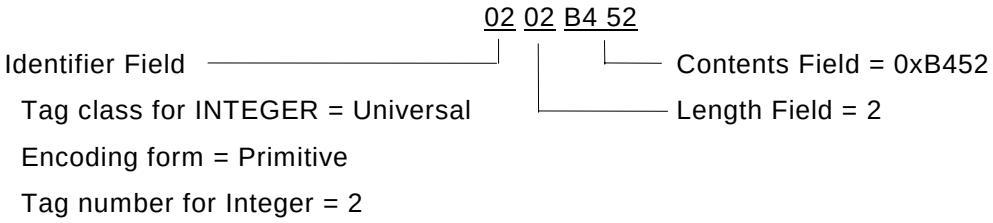


In BER, the tag class and number is conveyed by the Identifier field. When the tagging is EXPLICIT, the BER encoding of that value carries out two tags: the new one, specified within the square brackets and also the tag of the original, base type. The encoding is always constructed, with exactly one nested encoding, the encoding of the base type.

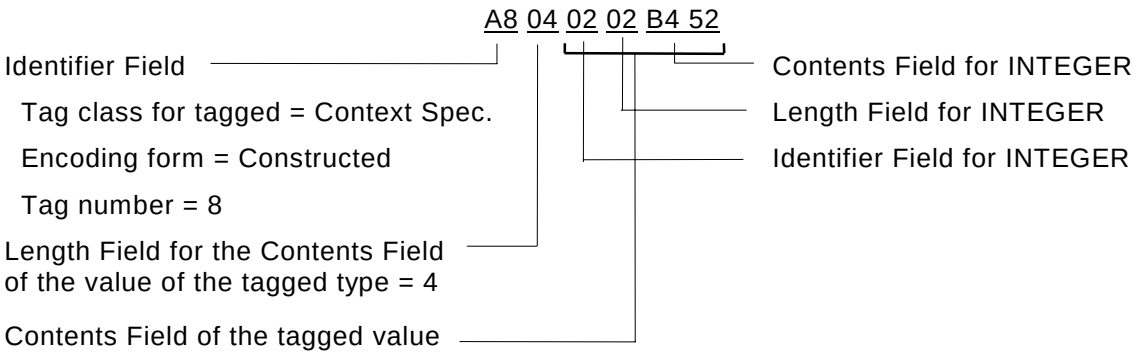
When IMPLICIT tagging is specified, the BER encoding of the tagged value contains only the new tag: it replaces the tag (class and number) of the original type.

Examples

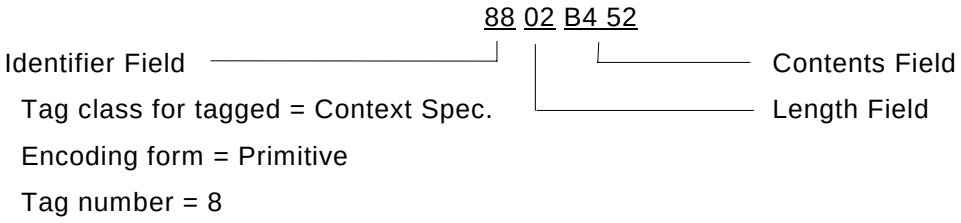
- BER encoding of the value –19374 (0xB452) of INTEGER type is as follows:



- BER encoding of the same value, but tagged as [8] INTEGER (explicitly tagged: it is the default tag style in BER) is as follows:



- BER encoding of the same value, but tagged as [8] IMPLICIT INTEGER (implicitly tagged) is as follows:



Pour A-XDR, la signification de l'étiquetage EXPLICITE et IMPLICITE est légèrement différente. Si A-XDR ne code pas le champ Identificateur pour des types non étiquetés du tout, ceci ne signifie pas que la nouvelle étiquette remplace l'ancienne. Par conséquent, *l'étiquetage implicite*, au sens de l'encodage BER, ne peut pas s'appliquer à A-XDR. Cependant, deux types d'étiquetage explicite peuvent être appliqués à A-XDR: *l'étiquetage explicite* (selon l'A-XDR) et *l'étiquetage explicite ASN.1.*

Pour l'A-XDR, *l'étiquetage explicite* correspond à la notation utilisée lorsque le numéro d'étiquette est explicitement présent dans la spécification ASN.1 pour assurer la distinction des composants dans les types SEQUENCE ou CHOICE ASN.1. Le codage de ces composants étiquetés dépend de leur nature: les étiquettes des composants CHOICE sont codées avec A-XDR comme des nombres entiers de longueur variable, représentant la valeur de l'étiquette. Cependant, les étiquettes des composants de SEQUENCE ne sont absolument pas codées. Si le codage A-XDR pour ces types d'A-XDR explicitement étiquetés est le même, que le mot-clef IMPLICITE soit présent dans la spécification ASN.1 ou non, on peut considérer que le mot-clef IMPLICITE est simplement ignoré pour l'A-XDR.

La notation du type, lorsque la classe et le numéro d'une étiquette sont explicitement spécifiés dans les spécifications ASN.1, par exemple [APPLICATION 30], est désignée par *étiquetage explicite ASN.1* pour l'A-XDR. Ce type d'étiquetage est seulement autorisé pour les éléments composés d'un type SEQUENCE.

Le codage A-XDR de valeurs de types explicitement étiquetés ASN.1 est le même que le codage BER mais l'utilisation de la forme indéfinie dans le champ Longueur n'est pas permise. L'objectif de cet étiquetage est juste de forcer le codage de la longueur, afin de permettre, par exemple, l'omission aisée de certaines structures.

6.8 Composants OPTIONAL et DEFAULT

Tout type composé ASN.1 peut comporter des composants désignés par les mots-clefs ASN.1 OPTIONAL et DEFAULT. Dans la spécification ASN.1, ces mots-clefs peuvent être placés après un type de composant et véhiculer une signification relativement intuitive. Les composants désignés par OPTIONAL peuvent être omis – la valeur du composant n'est pas toujours présente dans le codage. (Il convient que les circonstances et la signification de l'omission effective de ces composants soit spécifiée par le concepteur.)

En rendant un composant optionnel, il est possible de l'omettre lorsqu'il prend une certaine valeur – normalement celle qui est la plus fréquente. Ainsi, l'émission explicite de cette valeur peut être évitée. Le mot-clef DEFAULT ASN.1 désigne cette valeur par défaut pour un composant.

Noter que le fait de déclarer un composant OPTIONAL et de déclarer une valeur par DEFAULT pour celui-ci sont en général très différents (ces actions sont toutes deux exclusives en ASN.1). Alors que les composants OPTIONAL peuvent être totalement omis, les composants DEFAULT sont en fait toujours présents, car leur omission elle-même est utilisée pour représenter une valeur spécifique.

Le codage A-XDR d'un composant OPTIONAL ou DEFAULT commence avec la valeur codée d'un élément supplémentaire (en plus d'une syntaxe ASN.1 qui ne spécifie pas cet élément), nommé délimiteur d'utilisation. Ce délimiteur d'utilisation est de type BOOLEAN, sa valeur indique si la valeur du composant OPTIONAL ou DEFAULT est présente dans le codage ou non, comme suit:

- pour les composants OPTIONAL:

  délimiteur d'utilisation = TRUE Le composant est présent dans le codage

  délimiteur d'utilisation = FALSE Le composant n'est pas présent dans le codage

In A-XDR the meaning of EXPLICIT and IMPLICIT tagging is slightly different. Provided that A-XDR does not encode the Identifier field for non-tagged types at all, it has no means of saying that a new tag replaces the old one. Consequently, *Implicit tagging* in its BER meaning cannot be applied in A-XDR. On the other hand, two types of explicit tagging may be applied in A-XDR: *explicit tagging* (in terms of A-XDR) and *ASN.1 explicit tagging*.

In A-XDR we call *explicit tagging* the notation, when the tag number is explicitly present in the ASN.1 specification for ensuring distinctness of components in the ASN.1 CHOICE or SEQUENCE types. Encoding of these tagged components depends on their occurrence: tags of CHOICE components are encoded in A-XDR as variable-length integers, representing the value of the tag. On the other hand, tags of SEQUENCE components are not encoded at all. Provided that for these explicitly tagged types A-XDR encoding is the same, whether the IMPLICIT keyword is present in the ASN.1 specification or not, we can say that in A-XDR the IMPLICIT keyword is simply ignored.

The type notation when the class and the number of a tag is explicitly specified in the ASN.1 specification, for example, [APPLICATION 30], is referred to as *ASN.1 explicit tagging* in A-XDR. This type of tagging is allowed only for component members of a SEQUENCE type.

A-XDR encoding of values of ASN.1 explicit tagged types is the same as their BER encoding, except that using the indefinite form in the Length field is not allowed. The aim of this tagging is just to force the length to be encoded, in order to allow, for example, the easy omission of some structures.

6.8 OPTIONAL and DEFAULT components

Any ASN.1 composite type may contain components referenced with the ASN.1 keywords OPTIONAL or DEFAULT. In the ASN.1 specification, these keywords can be placed after a component type, and convey a quite intuitive meaning. Components marked OPTIONAL may be omitted – the component value is not always present in the encoding. (The circumstances and the significance of the actual omission of these components should be specified by the designer.)

One reason for making a component optional is to allow it to be omitted when it assumes a particular value – normally that which occurs most frequently. By this means, the explicit sending of that value can be avoided. The ASN.1 DEFAULT keyword designates such a default value for a component.

Note that declaring a component to be OPTIONAL and declaring a DEFAULT value for it are in general very different (they are mutually exclusive in ASN.1). While OPTIONAL components can be completely omitted, DEFAULT components are, in fact, always present, because even their omission is pressed into service to represent a specific value.

A-XDR encoding of an OPTIONAL or DEFAULT component begins with the encoded value of an additional element (additional to the ASN.1 syntax, which does not specify this element), called usage flag. This usage flag is typed BOOLEAN; its value indicates whether the value of the OPTIONAL or DEFAULT component is present in the encoding or not, as follows:

- for OPTIONAL components:

usage flag = TRUE	The component is present in the encoding
usage flag = FALSE	The component is not present in the encoding

- pour les composants DEFAULT:

délimiteur d'utilisation = TRUE	Le composant est présent dans le codage (une valeur différente que la valeur DEFAULT)
délimiteur d'utilisation = FALSE	Le composant n'est pas présent dans le codage (la valeur DEFAULT est véhiculée)

Si le délimiteur d'utilisation indique que le composant en question est présent dans le codage, le délimiteur d'utilisation est suivi par le codage A-XDR de la valeur du composant OPTIONAL ou DEFAULT. Cependant, lorsque le délimiteur d'utilisation indique que le composant n'est pas présent dans le codage, le codage A-XDR de ce composant prend fin avec le codage du délimiteur d'utilisation – aucun codage de la valeur du composant ne sera effectué.

Le paragraphe 6.9 et l'annexe C montrent des exemples de codage pour les composants OPTIONAL et DEFAULT.

6.9 Codage d'une valeur SEQUENCE

Comme le type CHOICE, le type SEQUENCE de l'ASN.1 est également défini comme un groupe de types de composants, tous distincts les uns des autres, mais contrairement au type CHOICE, chaque valeur d'un type SEQUENCE comporte une valeur de **chaque** type de composant[16]. L'ordre dans lequel il y a lieu que les valeurs du composant apparaissent dans le codage est fixe: il doit suivre l'ordre dans lequel les types de composant apparaissent dans la définition.

Le codage A-XDR d'une valeur SEQUENCE doit être le codage A-XDR d'une valeur de donnée de chaque type énoncé dans la définition ASN.1 du type SEQUENCE selon leur ordre d'apparition dans la définition, à moins que le type ait pour référence le mot-clef «OPTIONAL» ou «DEFAULT».

Les étiquettes de composants explicitement étiquetés d'une valeur SEQUENCE représentent une information redondante et ne sont donc pas codés. Le codage A-XDR d'une valeur de composant explicitement étiquetée est le codage A-XDR de la valeur du composant.

Lorsque le codage d'une valeur de donnée OPTIONAL ou DEFAULT est présent, il doit apparaître (codé comme spécifié en 6.8) dans le codage au point correspondant à l'apparence du type dans la définition ASN.1.

Exemples

Considérons la définition ASN.1 suivante:

```
Dummy_PDU ::= SEQUENCE {
    a       INTEGER(0..127),
    b       OCTET STRING(SIZE(4)) OPTIONAL,
    c       [1] BOOLEAN DEFAULT TRUE
}
```

16) A l'exception des types de composants marqués OPTIONAL ou DEFAULT (voir en 6.7 et 6.8).

- for DEFAULT components:

usage flag = TRUE	The component is present in the encoding (a different value than the DEFAULT value)
usage flag = FALSE	The component is not present in the encoding (the DEFAULT value is conveyed)

When the usage flag indicates that the component in question is present in the encoding, the usage flag is followed by the A-XDR encoding of the value of the OPTIONAL or DEFAULT component. On the other hand, when the usage flag indicates that the component is not present in the encoding, the A-XDR encoding of that component terminates with the encoding of the usage flag – no encoding of the component value will follow.

Subclause 6.9 and annex C show examples of encoding for OPTIONAL and DEFAULT components.

6.9 Encoding of a SEQUENCE value

Like the CHOICE type, the ASN.1 SEQUENCE type is also defined in terms of a collection of component types, all distinct, but contrary to the CHOICE type, each value of a SEQUENCE type contains a value of **each** of the component types[16]. The order in which the component values should appear in the encoding is fixed: it is the same order in which the component types appear in the definition.

A-XDR encoding of a SEQUENCE value shall be the A-XDR encoding of one data value from each of the types listed in the ASN.1 definition of the SEQUENCE type, in the order of their appearance in the definition, unless the type was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT".

Tags of explicitly tagged components of a SEQUENCE value represent redundant information, therefore are not encoded: A-XDR encoding of an explicit tagged component value is the A-XDR encoding of the component value.

When the encoding of an OPTIONAL or DEFAULT data value is present, it shall appear (encoded as specified in 6.8) in the encoding at the point corresponding to the appearance of the type in the ASN.1 definition.

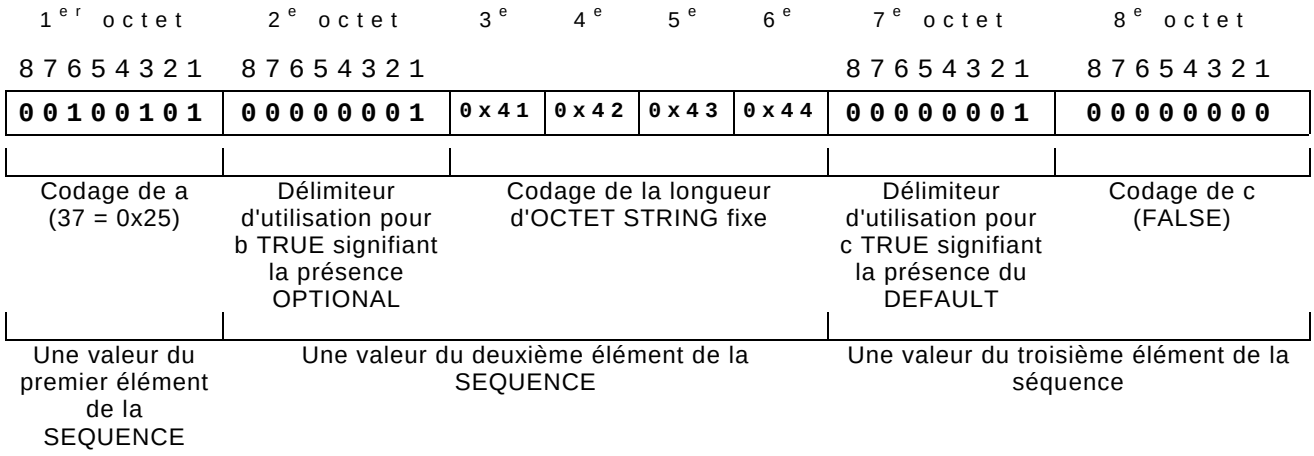
Examples

Taking the following ASN.1 definition:

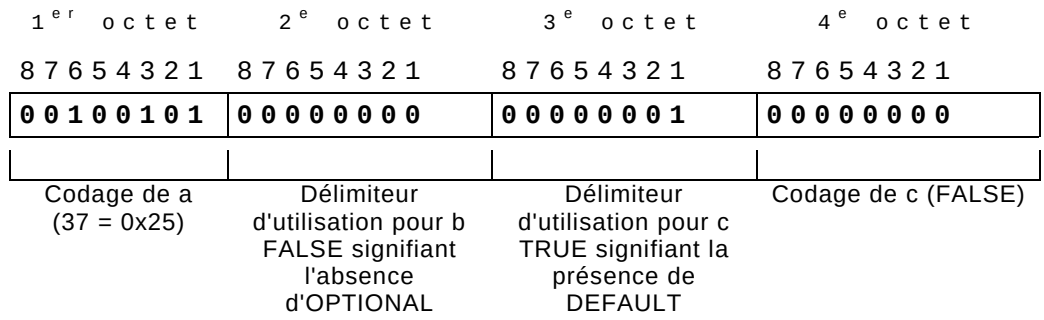
```
Dummy_PDU ::= SEQUENCE {
        a       INTEGER(0..127),
        b       BYTE STRING(SIZE(4)) OPTIONAL,
        c       [1] BOOLEAN DEFAULT TRUE
}
```

16) Except for component types marked OPTIONAL or DEFAULT (see 6.7 and 6.8).

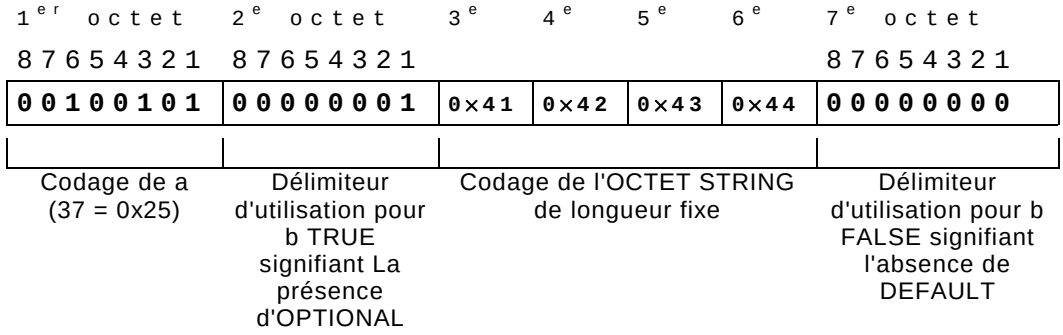
- Le codage A-XDR de la valeur de type SEQUENCE lorsque a = 37, b est présent et b = «ABCD» et c = FALSE est le suivant:



- Le codage A-XDR de la valeur de ce type SEQUENCE lorsque a = 37, b n'est pas présent et c = FALSE est le suivant:



- Le codage A-XDR de la valeur du type de SEQUENCE lorsque a = 37, b est présent et b = «ABCD», c = TRUE (transmission de sa valeur défaut) est la suivante:

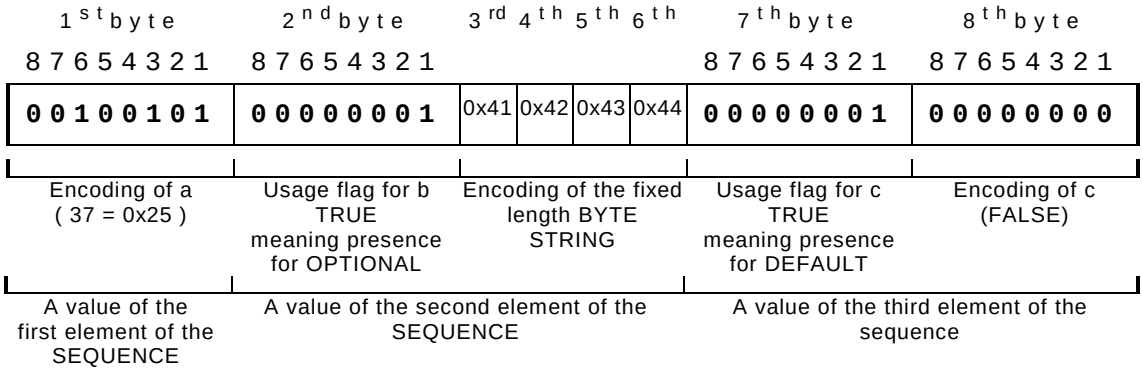


6.10 Codage d'une valeur SEQUENCE OF

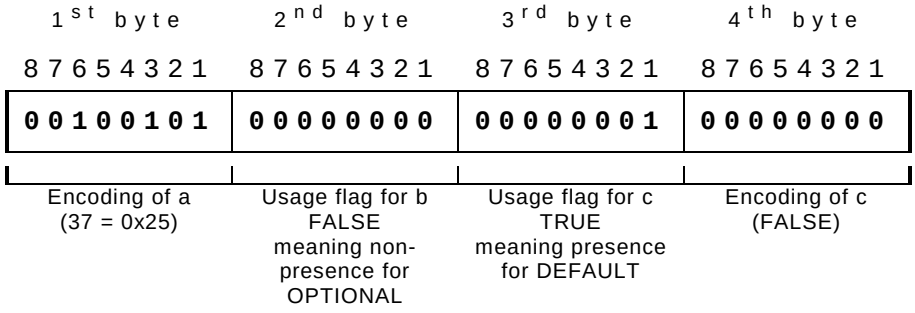
Le type SEQUENCE OF ASN.1 est défini comme un type de composant individuel et ses valeurs sont les groupes de valeurs du type de composant dans l'ordre.

A-XDR fournit deux types de codage pour le type SEQUENCE OF ASN.1, selon si la définition ASN.1 de la SEQUENCE OF spécifie la taille de la SEQUENCE OF ou non. Le codage en *longueur fixe* est utilisé lorsque la taille de la SEQUENCE OF est spécifiée dans la définition ASN.1 alors que les valeurs de SEQUENCE OF dont la taille n'est pas spécifiée sont codées en *longueur variable.*

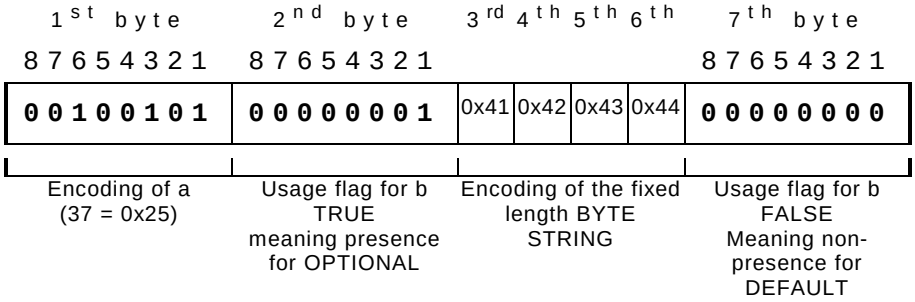
- A-XDR encoding of the value of that SEQUENCE type when a = 37, b is present and b = "ABCD" and c = FALSE is as follows:



- A-XDR encoding of the value of that SEQUENCE type when a = 37, b is not present and c = FALSE is as follows:



- A-XDR encoding of the value of that SEQUENCE type when a = 37, b is present and b = "ABCD", c = TRUE (it conveys its default value) is as follows:



6.10 Encoding of a SEQUENCE OF value

The ASN.1 SEQUENCE OF type is defined in terms of a single component type, and its values are the ordered collections of values of the component type.

A-XDR provides two types of encoding for the ASN.1 SEQUENCE OF type, depending on whether the ASN.1 definition of the SEQUENCE OF specifies the size of the SEQUENCE OF or not. *Fixed-length* encoding is provided when the size of a SEQUENCE OF is specified in the ASN.1 definition, while SEQUENCE OF values with no specified size are encoded in *variable-length* manner.

6.10.1 Codage en longueur fixe pour les valeurs de SEQUENCE OF dont la taille est spécifiée

Si la taille de la SEQUENCE OF est spécifiée dans la description ASN.1, le codage A-XDR de cette valeur de SEQUENCE OF contient seulement un champ Contenu. Les octets de ce champ doivent se composer du codage A-XDR de valeurs de donnée N selon le type mentionné dans la définition ASN.1, N étant la taille spécifiée. L'ordre du codage des valeurs de donnée doit être le même que l'ordre des valeurs de donnée dans la valeur de SEQUENCE OF à coder.

Exemple

Considérons la définition du type ASN.1 suivante:

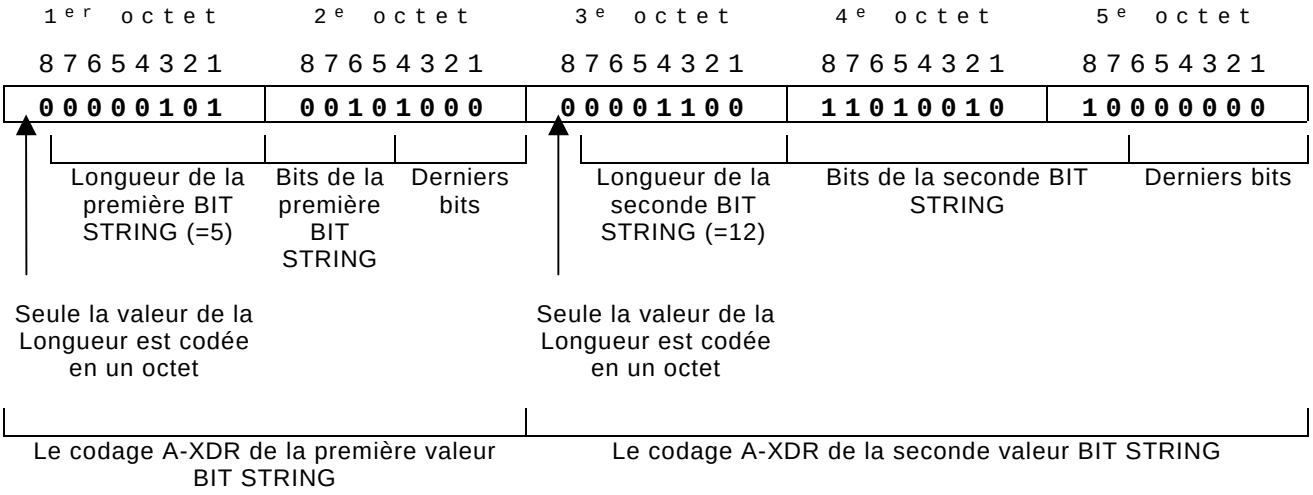
Dummy_List ::= SEQUENCE (SIZE(2)) OF BIT STRING

Les composants de la BIT STRING à coder sont les suivants:

première BIT STRING 00101

seconde BIT STRING 110100101000

Le codage A-XDR de la valeur ci-avant de ce type de SEQUENCE OF est le suivant:



6.10.2 Codage en longueur variable des valeurs SEQUENCE OF dont la taille n'est pas spécifiée

Le codage A-XDR de la valeur SEQUENCE OF lorsque la taille de la SEQUENCE OF n'est pas spécifiée dans les conditions ASN.1 comporte deux champs: un champ Longueur et un champ Contenu.

La valeur représentée par le champ Longueur est égale au nombre de composants dans la valeur codée du type SEQUENCE OF. Le champ Longueur lui-même est codé exactement de la même manière que celle spécifiée pour le codage du champ Longueur de la BIT STRING en longueur variable (6.4.2).

Le champ Contenu doit se composer du codage A-XDR de valeurs de donnée N selon le type mentionné dans la définition ASN.1, N'étant la valeur représentée par le champ Longueur. L'ordre du codage des valeurs de donnée doit être le même que celui des valeurs de donnée dans la valeur SEQUENCE OF à coder.

6.10.1 Fixed-length encoding for specified size SEQUENCE OF values

If the size of the SEQUENCE OF is specified in the ASN.1 description, the A-XDR encoding of that SEQUENCE OF value contains only a Contents field. The bytes of this field shall consist of the A-XDR encoding of N data values from the type listed in the ASN.1 definition, where N is the specified size. The order of the encoding of the data values shall be the same as the order of the data values in the SEQUENCE OF value to be encoded.

Example

Taking the following ASN.1 type definition:

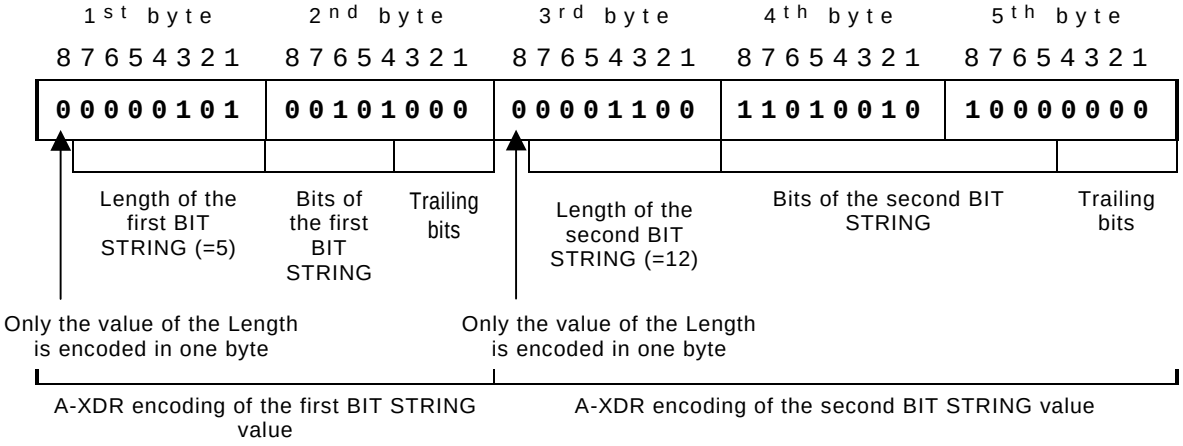
Dummy_List ::= SEQUENCE (SIZE(2)) OF BIT STRING

the component BIT STRINGS to be encoded are as follows:

first BIT STRING : 0 0 1 0 1

second BIT STRING : 1 1 0 1 0 0 1 0 1 0 0 0

A-XDR encoding of the above value of this SEQUENCE OF type is as follows:



6.10.2 Variable-length encoding for not specified size SEQUENCE OF values

A-XDR encoding of a SEQUENCE OF value when the size of the SEQUENCE OF is not specified in the ASN.1 definition contains two fields: a Length field and a Contents field.

The value represented by the Length field is equal to the number of components in the encoded value of the SEQUENCE OF type. The Length field itself is encoded exactly in the same way as specified for the Length field encoding of variable-length BIT STRINGS (6.4.2).

The Contents field shall consist of the A-XDR encoding of N data values from the type listed in the ASN.1 definition, where N is the value represented by the Length field. The order of the encoding of the data values shall be the same as the order of the data values in the SEQUENCE OF value to be encoded.

Exemple

Considérons la définition du type ASN.1 suivante:

Dummy_List ::= SEQUENCE OF INTEGER(0..4000)

Les deux composants à coder sont les suivants:

Integer 1 = 1956

Integer 2 = 3624

Le codage A-XDR de la valeur ci-avant de ce type de SEQUENCE OF est le suivant:

1er octet	2e octet	3e octet	4e octet	5e octet
8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1	8 7 6 5 4 3 2 1
0 0 0 0 0 0 1 0	**0 0 0 0 0 1 1 1**	**1 0 1 0 0 1 0 0**	**0 0 0 0 1 1 1 0**	**0 0 1 0 1 0 0 0**
Longueur (= 2 composants)	Codage du premier composant (nombre entier sans signe de longueur fixe 1956 = 07A4)		Codage du second composant (nombre entier sans signe de longueur fixe 3624 = 0x0E28)	

6.10.3 Codage du type SEQUENCE OF CHOICE spécial

Le SEQUENCE OF CHOICE est un cas spécial: bien que le type SEQUENCE OF soit défini comme un type de composant individuel, les éléments d'une structure de SEQUENCE OF CHOICE peuvent avoir différents types – les différentes alternatives du type CHOICE.

Le type de SEQUENCE OF CHOICE sera codé de la même manière que spécifié en 6.10.1 et en 6.10.2, mais le codage pour chaque composant de ce type SEQUENCE OF commencera avec le codage de l'étiquette du composant CHOICE sélectionné.

L'exemple 5 de l'annexe C donne des exemples pour le codage du SEQUENCE OF CHOICE.

6.11 Codage du type VisibleString

Le type VisibleString ASN.1 est un type de chaîne de caractères restreinte. Les chaînes de caractères restreintes sont des chaînes d'octets spéciales susceptibles de contenir des caractères provenant d'une série de caractères restreinte.

Même si l'ASN.1 spécifie huit types de cette sorte, seul le type VisibleString est utilisé dans la spécification DLMS (voir CEI 61334-4-41). Par conséquent A-XDR utilise seulement ce type[17] ASN.1.

L'ASN.1 attribue une étiquette – classe universelle – à chaque type de chaîne de caractères spécifique. Ces étiquettes – comme les étiquettes de tout autre type ASN.1 intégré – ne sont jamais codées en A-XDR. Par conséquent, le codage A-XDR de la VisibleString est le même que pour le type OCTET STRING. Si la CEI 61334-4-41 spécifie seulement des VisibleStrings de longueur variable, seul le codage de l'OCTET STRING de longueur variable peut être appliqué (voir en 6.5.2).

17) Néanmoins, la même méthode de codage peut également être appliquée aux autres types de chaînes de caractères restreintes ASN.1 également.

Example

Taking the following ASN.1 type definition:

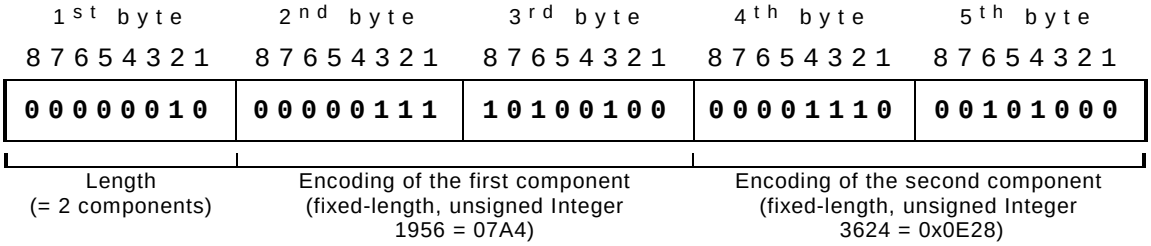
Dummy_List ::= SEQUENCE OF INTEGER(0..4000)

there are two components to be encoded as follows:

Integer 1 = 1956

Integer 2 = 3624

A-XDR encoding of the above value of this SEQUENCE OF type is as follows:



6.10.3 Encoding of the special SEQUENCE OF CHOICE type

The SEQUENCE OF CHOICE is a special case. Although the SEQUENCE OF type is defined in terms of a single component type, elements of a SEQUENCE OF CHOICE construction may have different types – the different alternatives of the CHOICE type.

The SEQUENCE OF CHOICE type will be encoded in the same manner, as described in 6.10.1 and 6.10.2, except that the encoding for each component in this SEQUENCE OF type will begin with the encoding of the tag of the selected CHOICE component.

Example 5 of annex C shows examples for SEQUENCE OF CHOICE encoding.

6.11 Encoding of the VisibleString type

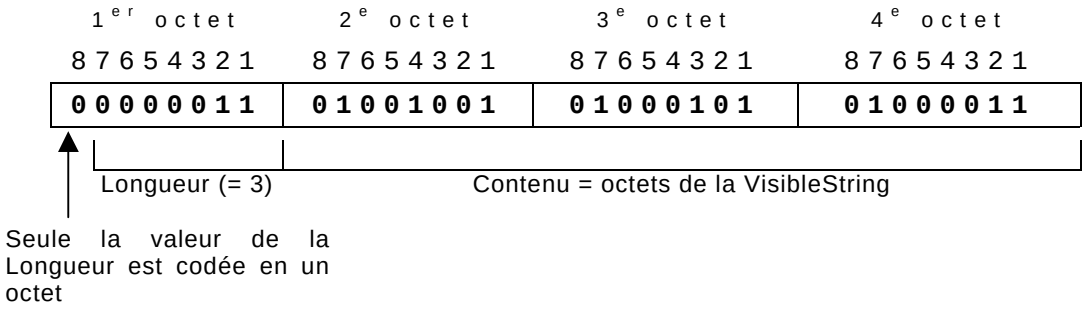
The VisibleString ASN.1 type is a restricted character string type. Restricted character strings are special byte strings which may contain characters drawn from a restricted character set. Although ASN.1 specifies eight such types, only the VisibleString type is used in the DLMS specification (see IEC 61334-4-41). Therefore, A-XDR provides support only for this ASN.1 type[17)].

ASN.1 assigns a Universal Class tag to each specific character string type. These tags, like tags of any other ASN.1 built-in type, are never encoded in A-XDR. Therefore, A-XDR encoding of the VisibleString is the same as if it were an BYTE STRING type. Provided that IEC 61334-4-41 specifies only variable-length VisibleStrings, only the variable length BYTE STRING encoding may be applied (see 6.5.2).

17) Nevertheless, the same encoding method may also be applied to the other ASN.1 restricted character string types.

Exemples

Le codage A-XDR de la valeur CEI d'une VisibleString est le suivant:



6.12 Codage du type de GeneralizedTime

Le GeneralizedTime est un Type Utile ASN.1, basé sur la VisibleString, spécifié comme suit:

GeneralizedTime ::= [UNIVERSAL 24] IMPLICIT VisibleString

Si A-XDR ne code jamais d'étiquettes ASN.1 de classe UNIVERSAL, le codage A-XDR du GeneralizedTime est – tout comme le codage VisibleString – le même que pour un type d'OCTET STRING. Pour les mêmes raisons que pour la VisibleString, le codage d'OCTET STRING de longueur variable s'applique pour le codage du GeneralizedTime (voir en 6.5.2).

6.13 Codage de la valeur/du type NULL ASN.1

Le type NULL ASN.1 est un type très spécial: il a juste une valeur. En raison de cette particularité, sa capacité de support d'information est relativement limitée, ce qui peut laisser à penser que le type NULL est inutile. Cependant, ce type s'avère utile dans un nombre de contextes limité: par exemple, lorsqu'un type doit être fourni mais qu'aucune information ne doit être transmise. Il est possible de considérer que le type NULL enrichit le domaine d'un autre type avec la valeur 'absente'.

Par exemple, le type suivant:

```
OutputValue ::= CHOICE {
    Known          [0] BOOLEAN,
    Unknown        [1] NULL
}
```

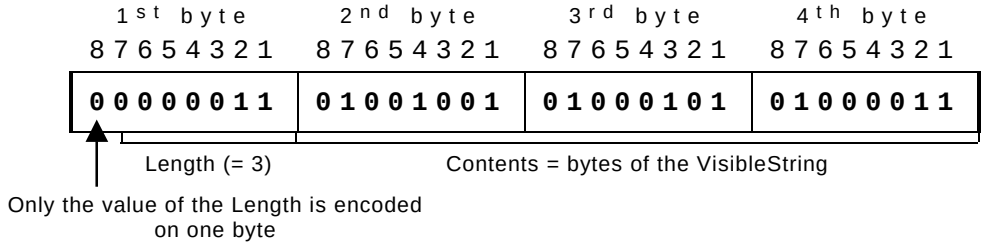
peut être utilisé si une valeur BOOLEAN doit être envoyée lorsqu'elle connue mais n'est parfois pas disponible.

Afin de pouvoir être codé avec l'A-XDR, il convient que le type NULL soit un type étiqueté.

Le codage A-XDR du type NULL est le codage A-XDR de la valeur de l'étiquette comme spécifié en 6.7.

Examples

A-XDR encoding of the IEC value of a VisibleString is as follows:



6.12 Encoding of the GeneralizedTime type

The GeneralizedTime is an ASN.1 Useful Type, based on the VisibleString, specified as follows:

GeneralizedTime ::= [UNIVERSAL 24] IMPLICIT VisibleString

Provided that A-XDR never encodes UNIVERSAL Class ASN.1 tags, A-XDR encoding of the GeneralizedTime is – similarly to the VisibleString encoding – the same as if it were an BYTE STRING type. For the same reason as for the VisibleString, the variable-length BYTE STRING encoding applies for the GeneralizedTime encoding (see 6.5.2).

6.13 Encoding of the ASN.1 NULL type/value

The ASN.1 NULL type is a very special type: it has just one value. Due to this speciality, its information-bearing capacity is somewhat limited. It may even be considered that the NULL type is useless. However, it proves to be useful in a small number of contexts, for example, where a type must be provided but no information needs to be conveyed. It can be considered that the NULL type enriches the domain of some other type with the 'absent' value.

For example, the following type

```
OutputValue::= CHOICE {
        Known       [0] BOOLEAN,
        Unknown     [1] NULL
}
```

can be used if a BOOLEAN value is to be sent when it is known, but sometimes it is not available.

In order to be able to be encoded with A-XDR, the NULL type should be a tagged type.

A-XDR encoding of the NULL type is the A-XDR encoding of the tag value as specified in 6.7.

Annexe A
(informative)

Extensibilité

Comme indiqué à l'article 3, contrairement à BER (Basic Encoding Rules), les règles de codage AXD-R ne sont pas extensibles. Quelles en sont les implications?

En considérant les SEQUENCES suivantes:

```
dummy_sequence_1    SEQUENCE
    {
        a       INTEGER,
        b       INTEGER
    }
dummy_sequence_2    SEQUENCE
    {
        a       INTEGER,
        c       BOOLEAN OPTIONAL,
        b       INTEGER
    }
```

le codage BER d'une valeur de ces séquences est identique en l'absence du composant OPTIONAL c. Cela signifie, avec l'encodage BER, que des composants optionnels supplémentaires peuvent être ajoutés aux séquences dans une version ultérieure d'un protocole d'application. Lorsque ces composants sont omis, par exemple pour communiquer avec un processus développé conformément à l'ancienne spécification, ce processus ne remarquera aucune différence. Cette propriété des règles de codage BER est appelée extensibilité.

A-XDR n'est pas extensible au sens du terme indiqué précédemment. En fait, comme il a été noté, même lorsque c est absent, le codage A-XDR comportera une référence à ce composant OPTIONAL.

En outre, A-XDR n'est pas fermé: il est non seulement capable de coder des types ASN.1 de la spécification DLMS actuelle, mais il permet également de spécifier un codage A-XDR pour les autres types ASN.1, comme REAL, SET OF, etc., si nécessaire.[1)]

[1)] L'actualisation d'A-XDR est sous la responsabilité du groupe de travail 9 du comité d'études 57.

Annex A
(informative)

Extensibility

As stated in clause 3, contrary to BER, A-XDR encoding rules are not extensible. What does this mean?

Taking the following SEQUENCES:

```
dummy_sequence_1 SEQUENCE
 {
        a       INTEGER,
        b       INTEGER
 }

dummy_sequence_2  SEQUENCE
 {
        a       INTEGER,
        c       BOOLEAN OPTIONAL,
        b       INTEGER
 }
```

BER encoding of a value of these sequences, when the 'c' OPTIONAL component is not present, is identical. This means that, with BER, extra optional components can be added to sequences in a future version of an application protocol. When these components are omitted, for example, for the purpose of communicating with a process developed according to the old specification, this process will notice no difference. This property of the BER encoding rules is called extensibility.

A-XDR is not extensible in the above way; in fact, as has been discussed, even when 'c' is not present, A-XDR encoding will contain a reference to this OPTIONAL component.

On the other hand, A-XDR is not closed. It is able to encode not only ASN.1 types of the current DLMS specification, but it is also possible to specify A-XDR encoding for other ASN.1 types, like REAL, SET OF, etc. if necessary.[1)]

1) Keeping A-XDR updated is the responsibility of working group 9 of IEC technical committee 57.

Annexe B
(informative)

Types et mots-clefs de l'ASN.1 utilisés en DLMS

Les types ASN.1 intégrés suivants sont utilisés pour la spécification DLMS (voir CEI 61334-4-41).

INTEGER
BOOLEAN
ENUMERATED
OCTET STRING
BIT STRING
VisibleString
CHOICE
SEQUENCE
SEQUENCE OF
NULL

CEI 61334-4-41 utilise également un type ASN.1 utile comme suit:

GeneralizedTime

Certains types utiles sont spécifiés par la spécification DLMS comme suit:

Integer8
Integer16
Integer32
Unsigned8
Unsigned16
Unsigned32

Et enfin, d'autres mots-clefs ASN.1 – qualificatifs – sont utilisés dans les spécifications DLMS comme suit:

IMPLICIT
OPTIONAL
DEFAULT

Annex B
(informative)

ASN.1 types and keywords used in DLMS

The following built-in ASN.1 types are used in the DLMS specification (see IEC 61334-4-41).

INTEGER
BOOLEAN
ENUMERATED
BYTE STRING
BIT STRING
VisibleString
CHOICE
SEQUENCE
SEQUENCE OF
NULL

IEC 61334-4-41 also uses an ASN.1 Useful Type, as follows:

GeneralizedTime

Some useful types are specified within the DLMS specification itself as follows:

Integer8
Integer16
Integer32
Unsigned8
Unsigned16
Unsigned32

Finally, some other ASN.1 keywords (qualifiers) are used in the DLMS specification as follows:

IMPLICIT
OPTIONAL
DEFAULT

Annexe C
(informative)

Exemples de codage A-XDR pour les PDU DLMS

Le type ASN.1 principal de PDU DLMS est CHOICE. Des exemples sont donnés uniquement pour des PDU non chiffrées, qui sont spécifiées comme suit:

```
DLMSpdu ::= CHOICE {

confirmedServiceRequest          [0]             ConfirmedServiceRequest,
initiateRequest                  [1] IMPLICIT    InitiateRequest,
getStatusRequest                 [2] IMPLICIT    GetStatusRequest,
getNameListRequest               [3] IMPLICIT    GetNameListRequest,
getVariableAttributeRequest      [4] IMPLICIT    GetVariableAttributeRequest
readRequest                      [5] IMPLICIT    ReadRequest,
writeRequest                     [6] IMPLICIT    WriteRequest,
confirmedServiceResponse         [7]             ConfirmedServiceResponse,
initiateResponse                 [8] IMPLICIT    InitiateResponse,
getStatusResponse                [9] IMPLICIT    GetStatusResponse,
getNameListResponse              [10] IMPLICIT   GetNameListResponse,
getVariableAttributeResponse     [11]            GetVariableAttributeResponse,
readResponse                     [12] IMPLICIT   ReadResponse,
writeResponse                    [13] IMPLICIT   WriteResponse,
confirmedServiceError            [14]            ConfirmedServiceError,
unconfirmedServiceRequest        [15]            UnconfirmedServiceRequest,
abortRequest                     [16] IMPLICIT   AbortRequest,
unconfirmedWriteRequest          [17] IMPLICIT   UnconfirmedWriteRequest,
unsolicitedServiceRequest        [18]            UnsolicitedServiceRequest,
informationReportRequest         [19] IMPLICIT   InformationReportRequest

Les autres choix correspondent aux PDU chiffrées

        .
        .
        .
        ded-informationReportRequest    [88] IMPLICIT    OCTET STRING
}
```

Annex C
(informative)

Examples of A-XDR encoding for DLMS PDUs

The outermost ASN.1 type of a DLMS PDU is CHOICE. Examples are given only for non-ciphered PDUs, which are specified as follows.

```
DLMSpdu ::=    CHOICE {
        confirmedServiceRequest             [0]             ConfirmedServiceRequest,
        initiateRequest                     [1] IMPLICIT    InitiateRequest,
        getStatusRequest                    [2] IMPLICIT    GetStatusRequest,
        getNameListRequest                  [3] IMPLICIT    GetNameListRequest,
        getVariableAttributeRequest         [4] IMPLICIT    GetVariableAttributeRequest,
        readRequest                         [5] IMPLICIT    ReadRequest,
        writeRequest                        [6] IMPLICIT    WriteRequest,
        confirmedServiceResponse            [7]             ConfirmedServiceResponse,
        initiateResponse                    [8] IMPLICIT    InitiateResponse,
        getStatusResponse                   [9] IMPLICIT    GetStatusResponse,
        getNameListResponse                 [10] IMPLICIT   GetNameListResponse,
        getVariableAttributeResponse        [11]            GetVariableAttributeResponse,
        readResponse                        [12] IMPLICIT   ReadResponse,
        writeResponse                       [13] IMPLICIT   WriteResponse,
        confirmedServiceError               [14]            ConfirmedServiceError,
        unconfirmedServiceRequest           [15]            UnconfirmedServiceRequest,
        abortRequest                        [16] IMPLICIT   AbortRequest,
        unconfirmedWriteRequest             [17] IMPLICIT   UnconfirmedWriteRequest,
        unsolicitedServiceRequest           [18]            UnsolicitedServiceRequest,
        informationReportRequest            [19] IMPLICIT   InformationReportRequest,

Other choices correspond to ciphered PDUs

        .
        .
        .
        ded-informationReportRequest        [88] IMPLICIT BYTE STRING
}
```

Exemple 1 – Codage A-XDR d'une PDU DLMS initiateRequest

Le type de InitiateRequest est spécifié comme suit:

InitiateRequest :: = SEQUENCE {

dedicated-key	OCTET STRING OPTIONAL,
response-allowed	BOOLEAN DEFAULT TRUE,
proposed-quality-of-service	[0] IMPLICIT Integer8 OPTIONAL,
proposed-dlms-version-number	Unsigned8,
proposed-conformance	Conformance,
proposed-max-pdu-size	Unsigned16

}

où la Conformance est spécifiée comme suit:

Conformance ::= [APPLICATION 30] IMPLICIT BIT STRING (SIZE(16))
{

get-data-set-attribute	(0),
get-ti-attribute	(1),
get-variable-attribute	(2),
read	(3),
write	(4),
unconfirmedWrite	(5),
change-scope	(6),
start	(7),
stop-resume	(8),
make-usable	(9),
data-set-load	(10),
selection-in-get-name-list	(11),
detailed-access-low-bit	(12),
detailed-access-high-bit	(13),
multiple-variable-list	(14),
data-set-upload	(15)

}

Example 1 – A-XDR encoding of an initiateRequest DLMS PDU

The InitiateRequest type is specified as follows:

```
InitiateRequest :: = SEQUENCE {
    dedicated-key                       BYTE STRING OPTIONAL,
    response-allowed                    BOOLEAN DEFAULT TRUE,
    proposed-quality-of-service         [0] IMPLICIT Integer8 OPTIONAL,
    proposed-dlms-version-number        Unsigned8,
    proposed-conformance                Conformance,
    proposed-max-pdu-size               Unsigned16
}
```

where Conformance could be as follows:

```
Conformance ::= [APPLICATION 30] IMPLICIT BIT STRING (SIZE(16))
{
    get-data-set-attribute          (0),
    get-ti-attribute                (1),
    get-variable-attribute          (2),
    read                            (3),
    write                           (4),
    unconfirmedWrite                (5),
    change-scope                    (6),
    start                           (7),
    stop-resume                     (8),
    make-usable                     (9),
    data-set-load                   (10),
    selection-in-get-name-list      (11),
    detailed-access-low-bit         (12),
    detailed-access-high-bit        (13),
    multiple-variable-list          (14),
    data-set-upload                 (15)
}
```

Pour les besoins de cet exemple, il faut prendre les valeurs suivantes en priorité:

dedicated-key	= absent
response-allowed	= TRUE (valeur par défaut)
proposed-quality-of-service	= présent, la valeur est 4
proposed-dlms-version-number	= 1
proposed-max-pdu-size	= 134 (= 0 x 86)
proposed-conformance	= 0x1C00, voir le tableau suivant:

B0	B1	B2	B3	B4	B5	B6	B7	B8	B8	B10	B11	B12	B13	B14	B15	
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	= 0x1C00[18)]

Le codage A-XDR de la PDU iniateRequest avec les valeurs précédentes est le suivant:

01	étiquette (étiquette explicite) de la PDU DLMS CHOICE (InitiateRequest)	
00	délimiteur d'utilisation pour le composant dedicated-key (FALSE, absent)	
00	délimiteur d'utilisation pour le composant response-allowed (FALSE, valeur par défaut véhiculée)	
01	délimiteur d'utilisation pour le composant proposed-quality-of-service (TRUE, présent)	
04	codage de la valeur du composant proposed-quality-of-service (4)	
01	codage du composant proposed-dlms-version-number (1)	
5E	codage de l'étiquette [APPLICATION 30] (étiquette explicite ASN.1)	Encodage BER du composant Conformance
03	longueur du champ 'contenu' en octets (3)	
00	nombre des bits non-utilisées dans le dernier octet du codage (0)	
10		
3C	encodage de la valeur du composant proposed-conformance	
00		
86	codage de la valeur du composant proposed-max-pdu-size (0x86)	

18) Cette valeur signifie qu'il est possible de supporter les services suivants (en plus des services obligatoires): read (bit 3), write (bit 4), et unconfirmed write (bit 5).

For the purpose of this example, the following values are to be taken first:

dedicated-key = not present
response-allowed = TRUE (default value)
proposed-quality-of-service = present, the value is 4
proposed-dlms-version-number = 1
proposed-max-pdu-size = 134 (= 0x86)
proposed-conformance = 0x1C00, as shown in the table below:

B0	B1	B2	B3	B4	B5	B6	B7	B8	B8	B10	B11	B12	B13	B14	B15	
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	= 0x1C00[18)]

A-XDR encoding of the initiateRequest PDU with the above values is as follows:

01	tag (explicit tag) of the DLMS PDU CHOICE (InitiateRequest)	
00	usage flag for the dedicated-key component (FALSE, not present)	
00	usage flag for the response-allowed component (FALSE, default value conveyed)	
01	usage flag for the proposed-quality-of-service component (TRUE, present)	
04	encoding of the value of the proposed-quality-of-service component (4)	
01	encoding of the proposed-dlms-version-number component (1)	
5E	encoding of the [APPLICATION 30] tag (ASN.1 explicit tag)	BER encoding of the Conformance Component
03	length of the 'contents' field in bytes (3)	
00	number of non-used bits in the last byte of the encoding (0)	
10		
3C	encoding the value of the proposed-conformance component	
00		
86	encoding of the value of the proposed-max-pdu-size component (0x86)	

18) This value means that the following services are proposed to be supported (besides the mandatory services): read (bit 3), write (bit 4), unconfirmed write (bit 5).

Exemple 2 – Codage A-XDR d'une PDU DLMS initiateResponse

Le type InitiateResponse est spécifié comme suit:

```
InitiateResponse ::= SEQUENCE
{
    negotiated-quality-of service      [0] IMPLICIT Integer8 OPTIONAL,
    negotiated-dlms-version-number     Unsigned8,
    negotiated-conformance             Conformance,
    negotiated-max-pdu-size            Unsigned16,
    vaa-name                           ObjectName
}
```

où la Conformance a la même spécification que dans l'exemple 1, et que l'ObjectName est spécifié comme:

```
ObjectName ::= Integer16
```

En outre, il convient que le nom vaa soit un nom vaa valide. Ceci signifie qu'il y a lieu que la valeur de classe de l'objet du nom vaa soit de 111 (il convient que les derniers trois bits significatifs du nom VAA Integer16 soient 111).

En prenant les mêmes valeurs pour les composants negotiated-quality-of-service, negotiated-DLMS-versions-number, negotiated-conformance et negotiated-max-PDU-size que dans l'exemple 1, la valeur pour le composant vaa-name 0x0037 a été choisie.

Le codage A-XDR de la PDU initiateResponse avec ces valeurs est le suivant:

```
08    étiquette (étiquette explicite) de la PDU DLMS CHOICE (InitiateResponse)
01    délimiteur d'utilisation pour le composant negotiated-quality-of-service (TRUE, présent)
04    codage de la valeur du composant negotiated-quality-of-service (4)
01    codage du composant negotiated-dlms-version-number (1)
5E    codage de l'étiquette [APPLICATION 30] (étiquette explicite ASN.1)   ┐ Codage BER
03    longueur du champ 'contenu' en octets (3)                            │ du
00    nombre des bits non-utilisées dans le dernier octet du codage (0)    │ composant
1C ┐                                                                       │ Conformance
00 ┘  codage du composant negotiated-conformance                           ┘
00 ┐
86 ┘  codage de la valeur du composant negotiated-max-pdu-size (0x86)
00 ┐
37 ┘  codage de la valeur du vaa-name (0x0037)
```

Example 2 – A-XDR encoding of an initateResponse DLMS PDU

The InitiateResponse type is specified as follows:

```
InitiateResponse ::= SEQUENCE
{
        negotiated-quality-of-service       [0] IMPLICIT Integer8 OPTIONAL,
        negotiated-dlms-version-number      Unsigned8,
        negotiated-conformance              Conformance,
        negotiated-max-pdu-size             Unsigned16,
        vaa-name                            ObjectName
}
```

where the Conformance has the same specification as in Example 1 and the ObjectName is specified as

```
ObjectName ::= Integer16
```

In addition, the vaa-name should be a valid vaa-name. This means that the object class value of the vaa-name should be 111 (the least significant three bits of the vaa-name Integer16 should be 111).

Taking the same values for the negotiated-quality-of-service, negotiated-DLMS-version-number, negotiated-conformance and negotiated-max-PDU-size as in Example 1, the value for the vaa-name component 0x0037 has been chosen.

A-XDR encoding of the initiateResponse PDU with these values is as follows:

```
08      tag (explicit tag) of the DLMS PDU CHOICE (InitiateResponse)
01      usage flag for the negotiated-quality-of-service component (TRUE, present)
04      encoding of the value of the negotiated-quality-of-service component (4)
01      encoding of the negotiated-dlms-version-number component (1)
5E      encoding of the [APPLICATION 30] tag (ASN.1 explicit tag)        ┐ BER encoding
03      length of the 'contents' field in bytes (3)                      │   of the
00      number of non-used bits in the last byte of the encoding (0)     │ Conformance
1C  ┐                                                                    │ Component
00  ┘   encoding the value of the negotiated-conformance component      ┘
00  ┐
86  ┘   encoding of the value of the negotiated-max-PDU-size component (0x86)
00  ┐
37  ┘   encoding of the value of the vaa-name (0x0037)
```

Exemple 3 – Codage A-XDR d'une PDU Confirmed Service Error

Cet exemple montre le codage A-XDR d'une PDU DLMS, généré lorsque le service InitiateRequest ne peut pas être accepté par le serveur DLMS, car la conformité proposée ne correspond pas à la liste de services du serveur donné.

La Confirmed Service Error est spécifiée comme suit:

```
ConfirmedServiceError ::= CHOICE
{
      -- tag 0 est réservée
      initiateError                 [1]     ServiceError,
      getStatus                     [2]     ServiceError,
      getNameList                   [3]     ServiceError,
      .                             .       .
      .                             .       .
      terminateUpLoad               [19]    ServiceError
}
```

où ServiceError est la suivante:

```
ServiceError ::=   CHOICE
{
      .                             .       .
      initiate                      [6]   IMPLICIT ENUMERATED
      -- initiate service error
      {

          autre                             (0),
          DLMS-version-too-low              (1),-- version DLMS proposée trop faible
          Incompatible-conformance          (2),-- services proposés insuffisants
          PDU-size-too-short                (3),-- taille de PDU proposée trop longue
          refused-by-the-VDE-Handler        (4)-- création de vaa impossible ou non autorisée
      }
      .                             .       .
      .                             .       .
}
```

Ainsi, le codage A-XDR de la PDU ConfirmedServiceError PDU avec les conditions précédentes est le suivant:

0E étiquette (explicite) de la PDU DLMS CHOICE (ConfirmedServiceError = 14)

01 étiquette (explicite) de ConfirmedServiceError CHOICE (initiateError = 1)

06 étiquette (explicite) de ServiceError CHOICE (initiate = 6)

02 codage de la valeur ENUMERATED (incompatible-conformance = 2)

Example 3 – A-XDR Encoding of a Confirmed Service Error PDU

This example shows A-XDR encoding of a DLMS PDU, which is generated when the InitateRequest service cannot be accepted by the DLMS server, because the proposed conformance does not fit the service list of the given server.

Confirmed Service Error is specified as follows:

```
ConfirmedServiceError ::= CHOICE
{
        -- tag 0 is reserved
        initiateError           [1]     ServiceError,
        getStatus               [2]     ServiceError,
        getNameList             [3]     ServiceError,
        .                       .       .
        .                       .       .
        terminateUpLoad         [19]    ServiceError
}
```

where ServiceError is as follows:

```
ServiceError ::= CHOICE
{
        .                       .       .
        initiate                [6]     IMPLICIT ENUMERATED
        -- initiate service error
        {
                other                           (0),
                DLMS-version-too-low            (1), -- proposed DLMS version too low
                incompatible-conformance        (2), -- proposed services not sufficient
                PDU-size-too-short              (3), -- proposed PDU size too long
                refused-by-the-VDE-Handler      (4)  -- vaa creation impossible or not allowed
        }
        .                       .       .
        .                       .       .
}
```

Therefore, A-XDR encoding of the ConfirmedServiceError PDU with the above conditions is as follows:

0E	tag (explicit) of the DLMS PDU CHOICE (ConfirmedServiceError = 14)
01	tag (explicit) of ConfirmedServiceError CHOICE (initiateError = 1)
06	tag (explicit) of ServiceError CHOICE (initiate = 6)
02	encoding of the ENUMERATED value (incompatible-conformance = 2)

Exemple 4 – Codage A-XDR des PDU Get Status Request / Get Status Response

Cet exemple montre le codage A-XDR d'une PDU Get Status Request et sa réponse, avec les conditions suivantes:

- La requête ne concerne pas les paramètres conditionnels (Identify = FALSE).
- L'instance VDE, qui reçoit cette requête, est spécifiée comme suit:

```
Dummy_VDE ::= VDE
{
        VDE-handler                 ,       --VDE-handler doit être défini ailleurs
        VDE-type        0x0001      ,       --valeur de type VDE (doit être définie ailleurs)
        Serial-Number   "1234"      ,       --doit être défini ailleurs
        Vendor-Name                 ,       --doit être défini par le fabricant
        Model                       ,       --doit être défini par le fabricant
        Version-Number              ,       --doit être défini par le fabricant
        Resources                   ,       --doit être défini ailleurs
        List-of-VAA     (7,15,23)   ,       --liste des vaa dans ce VDE
        Status          READY
}
```

La GetStatusRequest est spécifiée comme suit:

```
GetStatusRequest ::= Identify
```

où:

```
Identify ::= BOOLEAN
```

Ainsi, le codage A-XDR de la PDU GetStatusRequest PDU avec les conditions précédentes est le suivant:

02 étiquette (explicite) de la PDU DLMS CHOICE (GetStatusRequest = 2)

00 codage A-XDR de l'Identify BOOLEAN
(FALSE = y compris les paramètres obligatoires de la réponse)

Example 4 – A-XDR encoding of Get Status Request/Get Status Response PDUs

This example shows A-XDR encoding of a Get Status Request PDU and its response, with the following conditions:

- The request does not ask for conditional parameters (Identify = FALSE).
- The VDE instance which receives this request is specified as follows:

```
Dummy_VDE ::= VDE
{
        VDE-handler                ,      -- VDE-handler to be defined elsewhere
        VDE-type                   0x0001,  -- VDE type value (to be defined elsewhere)
        Serial-Number              "1234" ,  -- to be defined elsewhere
        Vendor-Name                ,      -- to be defined by the manufacturer
        Model                      ,      -- to be defined by the manufacturer
        Version-Number             ,      -- to be defined by the manufacturer
        Resources                  ,      -- to be defined elsewhere
        List-of-vaa                (7,15,23),  -- list of VAAs within this VDE
        Status                     READY
}
```

GetStatusRequest is specified as follows:

```
        GetStatusRequest ::= Identify
```

where

```
        Identify ::= BOOLEAN
```

Therefore, A-XDR encoding of the GetStatusRequest PDU with the above conditions is as follows:

02	tag (explicit) of the DLMS PDU CHOICE (GetStatusRequest = 2)
00	A-XDR encoding of the Identify BOOLEAN
	(FALSE = include only mandatory parameters in the response)

La GetStatusResponse est spécifiée comme suit:

```
GetStatusResponse ::= SEQUENCE

{
      vde-type              Integer16,
      serial-number         OCTET STRING,
      Status                ENUMERATED
                  {

                     ready                (0),
                     nochange             (1),
                     inoperable           (2)
                  } DEFAULT ready,

      list-of-vaa           SEQUENCE OF ObjectName,
      Identify              SEQUENCE
                  {
                       resources              VisibleString,
                       vendor-name            VisibleString,
                       model                  VisibleString,
                       version-number         Unsigned8
                  } OPTIONAL
}
```

Le codage A-XDR de la PDU GetStatusResponse, répondant à la requête GetStatusRequest avec Identify=FALSE, est le suivant:

```
09    étiquette (explicite) de la PDU DLMS CHOICE (GetStatusResponse)
00
01    codage de la valeur pour le VDE-Type (Integer16 = 0x0001)
04    longueur du Serial-Number (longueur variable) OCTET STRING (= 4)
31
32
33
34    Contenu du Serial-Number OCTET STRING (= "1234")
00    délimiteur d'utilisation pour le composant 'status': il est absent dans le codage19)
03    longueur de la list-of-vaa (longueur variable) SEQUENCE OF (= 3)
00
07    Premier VAA-Name de la liste (0x0007)
00
0F    Deuxième VAA-Name de la liste (15 = 0x000F)
00
17    Troisième VAA-Name de la liste (23 = 0x0017)
00    délimiteur d'utilisation pour la OPTIONAL Identify SEQUENCE (FALSE – absent)
```

19) C'est la valeur DEFAULT (prêt) qui est véhiculée par la PDU.

The GetStatusResponse is specified as follows:

```
GetStatusResponse ::=       SEQUENCE
{
        vde-type            Integer16,
        serial-number       BYTE STRING,
        status              ENUMERATED
                {
                        ready           (0),
                        nochange        (1),
                        inoperable      (2)
                } DEFAULT ready,
        list-of-vaa         SEQUENCE OF ObjectName,
        identify            SEQUENCE
                {
                        resources       VisibleString,
                        vendor-name     VisibleString,
                        model           VisibleString,
                        version-number  Unsigned8
                } OPTIONAL
}
```

The A-XDR encoding of the GetStatusResponse PDU responding to the GetStatusRequest request with Identify=FALSE is as follows:

```
09      tag (explicit tag) of the DLMS PDU CHOICE (GetStatusResponse)
00 ⎤
01 ⎦    encoding of the value for VDE-Type (Integer16 = 0x0001)
04      length of the Serial-Number (variable length) BYTE STRING (= 4)
31 ⎤
32 |
33 |
34 ⎦    Contents of the Serial-Number BYTE STRING (= "1234")
00      usage flag for the status component: it is not present in the encoding[19)]
03      length of the list-of-vaa (variable length) SEQUENCE OF (= 3)
00 ⎤
07 ⎦    First VAA-Name of the list (0x0007)
00 ⎤
0F ⎦    Second VAA-Name of the list (15 = 0x000F)
00 ⎤
17 ⎦    Third VAA-Name of the list (23 = 0x0017)
00      usage flag for the OPTIONAL Identify SEQUENCE (FALSE – not present)
```

19) It is the DEFAULT value (ready) which is conveyed by the PDU.

Exemple 5 – Services d'accès aux variables

Une Named Variable DLMS est spécifiée comme suit:

Dummy_NamedVariable ::= NamedVariable

{

Variable-Name	0x0010,
Scope-Of-Access	VDE-Specific,
Scope-May-Change	FALSE,
Lifetime	VDE,
Type-Description	Dummy_Type
Read-Write Flag	READ-WRITE
Available	TRUE

}

Dummy_Type ::= SEQUENCE

{

Number_Of_Counters	Unsigned8,
Counter_Values	SEQUENCE OF Unsigned16

}

Pour les besoins des exemples des services ReadRequest et ReadResponse, se référant à cette Dummy_NamedVariable, nous pouvons supposer que cette variable est dans notre VDE avec les valeurs suivantes:

Number_Of_Counters = 2

La Counters SEQUENCE OF contient deux valeurs comme suit:

Counter_Value_1	= 318 (= 0x013E)
Counter_Value_2	= 715 (= 0x02CB)

Cette Named Variable sera utilisée dans les exemples suivants.

Example 5 – Variable Access Services

A DLMS Named Variable is specified as follows:

```
Dummy_NamedVariable ::=   NamedVariable
{
        Variable-Name           0x0010,
        Scope-Of-Access         VDE-specific,
        Scope-May-Change        FALSE,
        Lifetime                VDE,
        Type-Description        Dummy_Type,
        Read-Write Flag         READ-WRITE
        Available               TRUE
}
Dummy_Type ::=   SEQUENCE
{
        Number_Of_Counters  Unsigned8,
        Counter_Values          SEQUENCE OF Unsigned16
}
```

For the purpose of examples for ReadRequest and ReadResponse services, referencing to this Dummy_NamedVariable, we can suppose that we have this variable within our VDE with the following values:

Number_Of_Counters = 2

The Counters SEQUENCE OF contains two values as follows:

Counter_Value_1 = 318 (= 0x013E)
Counter_Value_2 = 715 (= 0x02CB)

This Named Variable will be used in the following examples.

Exemple 5.1 – Lecture d'une Named Variable sans accès détaillé

Le ReadRequest Service est spécifié comme suit:

```
ReadRequest ::=          SEQUENCE OF VariableAccessSpecification
```

où:

```
VariableAccessSpecification ::=     CHOICE
{
      Variable-name          [2]                    IMPLICIT ObjectName,
      Detailed-access        [3]                    IMPLICIT SEQUENCE

                  {
                        variable-name          ObjectName
                        detailed-access        Detailed-access
                  }

}
```

Pour le moment, l'accent est mis sur l'accès sans accès détaillé. Le codage A-XDR de la PDU ReadRequest, demandant la valeur de la Dummy_NamedVariable, est donc le suivant:

```
05          étiquette (explicite) de la PDU DLMS CHOICE (ReadRequest = 15)
01          longueur de la (longueur variable) SEQUENCE OF (=1)
02          étiquette (explicite) de la VariableAccessSpecification CHOICE (variable-name = 2)
00
10          codage de la référence de variable ObjectName (VariableName) = 0x0010)
```

La réponse à cette requête – si la requête peut être acceptée – est un message ReadResponse. Il est spécifié comme suit:

```
ReadResponse ::=          SEQUENCE OF CHOICE {

   data                   [0] Data
   data-access-error      [1] IMPLICIT DataAccessError

}
```

où

```
Data ::=          CHOICE{
   array                  [1] IMPLICIT   SEQUENCE OF Data,
   structure              [2] IMPLICIT   SEQUENCE OF Data,
   .                      .              .
   unsigned               [17] IMPLICIT Unsigned8,
   long-unsigned          [18] IMPLICIT Unsigned16,20)
}
```

20) Pour l'instant, nous n'avons besoin que de ces CHOICE.

Example 5.1 – Read a Named Variable without detailed access

The ReadRequest Service is specified as follows:

```
    ReadRequest ::=   SEQUENCE OF VariableAccessSpecification
where
    VariableAccessSpecification ::=   CHOICE
{
        variable-name          [2]     IMPLICIT ObjectName,
        detailed-access        [3]     IMPLICIT SEQUENCE
                        {
                                variable-name          ObjectName,
                                detailed-access        DetailedAccess
                        }
}
```

For the moment, the focus is on access without detailed access. A-XDR encoding of the ReadRequest PDU, asking for the value of the Dummy_NamedVariable is, thus, as follows:

```
05      tag (explicit) of the DLMS PDU CHOICE (ReadRequest = 15)
01      length for the (variable length) SEQUENCE OF (=1)
02      tag (explicit) of VariableAccessSpecification CHOICE (variable-name = 2)
00 ⎤
10 ⎦    encoding of the variable reference ObjectName (VariableName = 0x0010)
```

The response for this request (when the request can be accepted) is a ReadResponse message. It is specified as follows:

```
ReadResponse ::=      SEQUENCE OF CHOICE {
        data                    [0] Data,
        data-access-error       [1] IMPLICIT    DataAccessError
}
where

Data ::=        CHOICE{
        array                   [1] IMPLICIT    SEQUENCE OF Data,
        structure               [2] IMPLICIT    SEQUENCE OF Data,
        .                       .               .
        unsigned                [17] IMPLICIT   Unsigned8,
        long-unsigned           [18] IMPLICIT   Unsigned16,}[20)]
```

[20)] For the moment, we need only these CHOICEs.

Le codage A-XDR de la PDU ReadResponse contenant la Dummy_NamedVariable avec les valeurs données est le suivant:

0C étiquette (explicite) de la PDU DLMS CHOICE (ReadResponse = 12)

01 longueur de la longueur variable SEQUENCE OF CHOICE de ReadResponse (1)

00 étiquette (explicite) du CHOICE (l'élément sélectionné est [0] Data)

02 étiquette (explicite) du CHOICE of Data (l'élément sélectionné est [2] structure)

02 longueur de la longueur variable SEQUENCE OF Data (CHOICE)
(la structure a deux membres)

11 étiquette (explicite) du premier composant de la structure [17] Unsigned8)

02 valeur codée de Number_Of_Counters (= 2)

01 étiquette (explicite) du deuxième composant de la structure ([1], tableau)

02 longueur du tableau (SEQUENCE OF Data (CHOICE))
(= 2, il y a deux Data dans le tableau)

12 étiquette (explicite) de la première Data ([18] Unsigned16)

01

3E valeur codée du Counter_Value_1 (=0x013E)

12 étiquette (explicite) de la deuxième Donnée ([18] Unsigned16)

02

CB valeur codée de Counter_Value_2 (=0x02CB)

A-XDR encoding of the ReadResponse PDU containing the Dummy_NamedVariable with the given values is as follows:

0C tag (explicit tag) of the DLMS PDU CHOICE (ReadResponse = 12)

01 length for the variable length SEQUENCE OF CHOICE of ReadResponse (1)

00 tag (explicit) of the CHOICE (the selected element is [0] Data)

02 tag (explicit) of the CHOICE of Data (the selected element is [2] structure)

02 length for the variable length SEQUENCE OF Data (CHOICE)
(the structure has two members)

11 tag (explicit) of the first component of the structure ([17] Unsigned8)

02 encoded value of Number_Of_Counters (= 2)

01 tag (explicit) of the second component of the structure ([1], array)

02 length of the array (SEQUENCE OF Data (CHOICE))
(= 2, there are two Data in the array)

12 tag (explicit) of the first Data ([18] Unsigned16)

01

3E encoded value of Counter_Value_1 (=0x013E)

12 tag (explicit) of the second Data ([18] Unsigned16)

02

CB encoded value of Counter_Value_2 (=0x02CB)

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 2-8318-5262-5

ICS 29.240.20; 33.200

Typeset and printed by the IEC Central Office
GENEVA, SWITZERLAND