UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T X.691

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(04/95)

RÉSEAUX DE COMMUNICATION DE DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

RÉSEAUTAGE OSI ET ASPECTS DES SYSTÈMES –

NOTATION DE SYNTAXE ABSTRAITE NUMÉRO UN (ASN.1)

TECHNOLOGIES DE L'INFORMATION – RÈGLES DE CODAGE ASN.1 – SPÉCIFICATION DES RÈGLES DE CODAGE COMPACT

Recommandation UIT-T X.691
Remplacée par une version plus récente

(Antérieurement «Recommandation du CCITT»)

Remplacée par une version plus récente

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Au sein de l'UIT-T, qui est l'entité qui établit les normes mondiales (Recommandations) sur les télécommunications, participent quelque 179 pays membres, 84 exploitations de télécommunications reconnues, 145 organisations scientifiques et industrielles et 38 organisations internationales.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution nº 1 de la Conférence mondiale de normalisation des télécommunications (CMNT) (Helsinki, 1993). De plus, la CMNT, qui se réunit tous les quatre ans, approuve les Recommandations qui lui sont soumises et établit le programme d'études pour la période suivante.

Dans certains secteurs de la technologie de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI. Le texte de la Recommandation X.691 de l'UIT-T a été approuvé le 10 avril 1995. Son texte est publié, sous forme identique, comme Norme internationale ISO/CEI 8825-2.

NOTE

Dans la présente Recommandation, l'expression «Administration» est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

Remplacée par une version plus récente

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX DE COMMUNICATION DE DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

(Février 1994)

ORGANISATION DES RECOMMANDATIONS DE LA SÉRIE X

Domaine	Recommandations
RÉSEAUX PUBLICS POUR DONNÉES	
Services et services complémentaires	X.1-X.19
Interfaces	X.20-X.49
Transmission, signalisation et commutation	X.50-X.89
Aspects réseau	X.90-X.149
Maintenance	X.150-X.179
Dispositions administratives	X.180-X.199
INTERCONNEXION DES SYSTÈMES OUVERTS	
Modèle et notation	X.200-X.209
Définition des services	X.210-X.219
Spécifications des protocoles en mode connexion	X.220-X.229
Spécifications des protocoles en mode sans connexion	X.230-X.239
Formulaires PICS	X.240-X.259
Identification des protocoles	X.260-X.269
Protocoles de sécurité	X.270-X.279
Objets gérés de couche	X.280-X.289
Test de conformité	X.290-X.299
INTERFONCTIONNEMENT DES RÉSEAUX	
Considérations générales	X.300-X.349
Systèmes mobiles de transmission de données	X.350-X.369
Gestion	X.370-X.399
SYSTÈMES DE MESSAGERIE	X.400-X.499
ANNUAIRE	X.500-X.599
RÉSEAUTAGE OSI ET ASPECTS DES SYSTÈMES	
Réseautage	X.600-X.649
Dénomination, adressage et enregistrement	X.650-X.679
Notation de syntaxe abstraite numéro un (ASN.1)	X.680-X.699
GESTION OSI	X.700-X.799
SÉCURITÉ	X.800-X.849
APPLICATIONS OSI	
Engagement, concomitance et rétablissement	X.850-X.859
Traitement des transactions	X.860-X.879
Opérations distantes	X.880-X.899
TRAITEMENT OUVERT RÉPARTI	X.900-X.999

Remplacée par une version plus récente

TABLE DES MATIÈRES

Résumé

La présente Recommandation | Norme internationale décrit un ensemble de règles de codage applicables aux valeurs de tous les types ASN.1. Ces règles donnent une représentation plus compacte que celle que l'on peut obtenir au moyen des règles de codage de base et de leurs dérivées (décrites dans la Recommandation X.690).

Introduction

L'ensemble de documents Rec. UIT-T X.680 | ISO/CEI 8824-1, Rec. UIT-T X.681 | ISO/CEI 8824-2, Rec. UIT-T X.682 | ISO/CEI 8824-3, Rec. UIT-T X.683 | ISO/CEI 8824-4, Rec. UIT-T X.680/Amd. 1 | ISO/CEI 8824-1/Amd. 1, Rec. UIT-T X.681/Amd. 1 | ISO/CEI 8824-2/Amd. 1, décrivent la notation de syntaxe abstraite numéro un (ASN.1) qui permet de définir les messages échangés par des applications homologues.

La présente Recommandation | Norme internationale définit les règles de codage qui pourront être appliquées à des valeurs de types définis conformément à la notation spécifiée dans la Rec. UIT-T X.680 | ISO/CEI 8824-1. L'application de ces règles de codage produit une syntaxe de transfert pour de telles valeurs. La spécification de ces règles de codage postule implicitement que ces règles pourront être utilisées telles quelles pour le décodage.

Plusieurs ensembles de règles de codage peuvent être appliqués à des valeurs de types ASN.1. La présente Recommandation | Norme internationale définit un ensemble de règles de codage compact (PER) (*packed encoding rules*), ainsi dénommées parce qu'elles donnent une représentation plus compacte que celle que l'on peut obtenir au moyen des règles de codage de base (BER) et de leurs dérivées, décrites dans la Rec. UIT-T X.690 | ISO/CEI 8825-1, à laquelle font référence certaines parties de la spécification des présentes règles de codage compact.

NORME INTERNATIONALE

RECOMMANDATION UIT-T

TECHNOLOGIES DE L'INFORMATION – RÈGLES DE CODAGE ASN.1 – SPÉCIFICATION DES RÈGLES DE CODAGE COMPACT

1 Domaine d'application

La présente Recommandation | Norme internationale spécifie un ensemble de règles de codage compact qui peuvent être utilisées pour élaborer une syntaxe de transfert applicable à des valeurs de types définis dans la Rec. UIT-T X.680 | ISO/CEI 8824-1. Ces règles de codage compact sont également applicables au décodage d'une telle syntaxe de transfert afin d'identifier les valeurs de données qui sont transférées.

Les règles de codage spécifiées dans la présente Recommandation | Norme internationale:

- sont utilisées au moment de la communication;
- sont destinées à être utilisées dans des circonstances où la minimisation du volume occupé par la représentation de valeurs est la principale préoccupation lors du choix de règles de codage;
- permettent l'extension d'une syntaxe abstraite par adjonction de valeurs supplémentaires, tout en conservant les codages des valeurs existantes, pour toutes les formes d'extension décrites dans la Rec. UIT-T X.680/Amd. 1 | ISO/CEI 8824-1/Amd. 1.

2 Références normatives

Les Recommandations et les Normes internationales suivantes contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente Recommandation | Norme internationale. Au moment de la publication, les éditions indiquées étaient en vigueur. Toutes Recommandations et Normes sont sujettes à révision et les parties prenantes aux accords fondés sur la présente Recommandation | Norme internationale sont invitées à rechercher la possibilité d'appliquer les éditions les plus récentes des Recommandations et Normes indiquées ci-après. Les membres de la CEI et de l'ISO possèdent le registre des Normes internationales en vigueur. Le Bureau de la normalisation des télécommunications de l'UIT tient à jour une liste des Recommandations de l'UIT-T en vigueur.

2.1 Recommandations | Normes internationales identiques

- Recommandation UIT-T X.200 (1994) | ISO/CEI 7498-1:1994, *Technologies de l'information – Interconnexion des systèmes ouverts – Modèle de référence de base: le modèle de référence de base.*
- Recommandation UIT-T X.216 (1994) | ISO/CEI 8822:1994, *Technologies de l'information – Interconnexion des systèmes ouverts – Définition du service de présentation.*
- Recommandation UIT-T X.226 (1994) | ISO/CEI 8823-1:1994, *Technologies de l'information – Interconnexion des systèmes ouverts – Protocole de présentation en mode connexion: spécification du protocole.*
- Recommandation UIT-T X.680 (1994) | ISO/CEI 8824-1:1995, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification de la notation de base.*
- Recommandation UIT-T X.680 (1994)/Amd. 1 (1995) | ISO/CEI 8824-1:1995/Amd. 1:1995, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification de la notation de base – Amendement 1: règles d'extensibilité.*
- Recommandation UIT-T X.681 (1994) | ISO/CEI 8824-2:1995, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification des objets informationnels.*
- Recommandation UIT-T X.681 (1994)/Amd. 1 (1995) | ISO/CEI 8824-2:1995/Amd. 1:1995, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification des objets informationnels – Amendement 1: règles d'extensibilité.*

- Recommandation UIT-T X.682 (1994) | ISO/CEI 8824-3:1995, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification des contraintes.*
- Recommandation UIT-T X.683 (1994) | ISO/CEI 8824-4:1995, *Technologies de l'information – Notation de syntaxe abstraite numéro un: paramétrage des spécifications de la notation de syntaxe abstraite numéro un.*
- Recommandation UIT-T X.690 (1994) | ISO/CEI 8825-1:1995, *Technologies de l'information – Règles de codage de la notation de syntaxe abstraite numéro un: spécification des règles de codage de base, des règles de codage canoniques et des règles de codage distinctives.*

2.2 Autres références

- Recommandation X.208 du CCITT (1988), *Spécification de la syntaxe abstraite numéro un (ASN.1).*
- ISO/CEI 2022:1994, *Technologies de l'information – Structure de code de caractères et techniques d'extension.*
- ISO 2375:1985, *Traitement de l'information – Procédure pour l'enregistrement des séquences d'échappement.*
- ISO 6093:1985, *Traitement de l'information – Représentation des valeurs numériques dans les chaînes de caractères pour l'échange d'information.*
- ISO/CEI 8824:1990, *Technologies de l'information – Interconnexion de systèmes ouverts – Spécification de la notation de syntaxe abstraite numéro un (ASN.1).*
- ISO *Registre international des jeux de caractères codés à utiliser avec une séquence d'échappement.*
- ISO/CEI 10646-1:1993, *Technologies de l'information – Jeu universel de caractères codés à plusieurs octets – Partie 1: Architecture et table multilingue.*

3 Définitions

Pour les besoins de la présente Recommandation | Norme internationale, les définitions suivantes s'appliquent.

3.1 Définition du service de présentation de base

Termes définis dans la Rec. UIT-T X.216 | ISO/CEI 8822:

a) ensemble de contextes défini;

b) identificateur de contexte de présentation.

3.2 Spécification de la notation de base

Toutes les définitions contenues dans la Rec. UIT-T X.680 | ISO/CEI 8824-1 s'appliquent.

3.3 Extensibilité de la notation ASN.1

Termes définis dans la Rec. UIT-T X.680/Amd. 1 | ISO/CEI 8824-1/Amd. 1:

a) marqueur d'extension;

b) arc d'extension;

c) racine d'extension;

d) prolongement d'extension.

3.4 Spécification des objets informationnels

Toutes les définitions figurant dans la Rec. UIT-T X.681 | ISO/CEI 8824-2 s'appliquent.

3.5 Spécification des contraintes

Termes définis dans la Rec. UIT-T X.682 | ISO/CEI 8824-3:

a) contrainte relationnelle de composante;

b) contrainte tabulaire.

3.6 Spécification du paramétrage en notation ASN.1

Terme défini dans la Rec. UIT-T X.683 | ISO/CEI 8824-4:

– contrainte variable.

3.7 Règles de codage de base

Termes définis dans la Rec. UIT-T X.690 | ISO/CEI 8825-1:

a) conformité dynamique;

b) conformité statique;

c) valeur de données;

d) codage (d'une valeur de données);

e) expéditeur;

f) destinataire.

3.8 Autres définitions

De plus, les définitions suivantes s'appliquent.

3.8.1 codage d'entier binaire en complément à 2: codage d'un nombre entier sur un champ binaire aligné à l'octet de longueur spécifiée, ou sur le nombre minimal d'octets permettant de représenter cet entier (égal, supérieur ou inférieur à zéro comme spécifié au 10.4) sous forme d'un entier en complément à deux.

NOTES

1 La représentation d'un nombre binaire en complément à deux est obtenue en numérotant les bits des octets qui le composent, en commençant par le bit 1 du dernier octet qui devient le bit 0 et en terminant par le bit 8 du premier octet. A chaque bit est affectée une valeur numérique de 2^N, N étant la position du bit dans la séquence de numérotation précédente. La valeur du nombre binaire en complément à deux est obtenue en ajoutant les valeurs numériques affectées à chacun des bits qui sont à un, sauf le bit 8 du premier octet, puis en soustrayant de cette valeur la valeur numérique affectée à ce bit 8 du premier octet, s'il est à un.

2 L'expression *nombre entier* est synonyme du terme mathématique *entier*. Elle est utilisée à la place de celui-ci pour éviter une confusion avec le type entier (*integer*) de l'ASN.1.

3.8.2 valeur d'une syntaxe abstraite: valeur d'une syntaxe abstraite (définie comme l'ensemble des valeurs d'un type unique ASN.1), à coder selon les règles PER, ou à générer par un décodage PER.

NOTE – Le type ASN.1 unique associé à une syntaxe abstraite est identifié de façon formelle par un objet de la classe ABSTRACT-SYNTAX.

3.8.3 champ binaire: produit d'une partie du processus de codage, qui se compose d'un ensemble ordonné d'éléments binaires. Cet ensemble n'est pas nécessairement un multiple de 8 et ne commence pas nécessairement à une limite d'octet dans le codage complet de la valeur de la syntaxe abstraite.

3.8.4 codage canonique: codage complet d'une valeur dans la syntaxe abstraite, obtenu par application de règles de codage ne comportant aucune option dépendant de la mise en œuvre; de telles règles se traduisent – dans la syntaxe de transfert et dans les valeurs de la syntaxe abstraite – par des correspondances biunivoques entre chaînes binaires non ambiguës et uniques de la syntaxe de transfert et valeurs de la syntaxe abstraite.

3.8.5 type composite: type du genre ensemble, séquence, ensemble-de, séquence-de, choix, valeur PDV encapsulée, externe ou chaîne de caractères.

3.8.6 valeur composite: valeur d'un type composite.

3.8.7 nombre entier contraint: nombre entier soumis par les contraintes visibles des règles PER, de manière à s'inscrire dans un intervalle compris entre une borne inférieure "lb" et une borne supérieure "ub", bornes comprises, avec "lb" inférieure ou égale à "ub".

NOTE – Les nombres entiers contraints apparaissent dans les codages qui identifient l'alternative choisie dans un type choix, ou la longueur d'une chaîne binaire, de caractères ou d'octets lorsque la longueur du type de celle-ci est limitée à un maximum par une contrainte visible des règles PER, ou le nombre de composantes d'une valeur de type séquence-de ou ensemble-de lorsque le nombre des composantes d'un tel type est limité à un maximum par une contrainte visible des règles PER, ou la valeur d'un entier lorsque le type de celui-ci est limité à un intervalle fini par une contrainte visible des règles PER, ou la valeur ordinale d'un élément appartenant à un type énuméré.

3.8.8 contrainte effective de taille (pour un type chaîne contrainte): contrainte unique limitant une taille à une valeur finie, qui peut être appliquée à un type chaîne natif et dont l'effet sera de permettre toutes les longueurs – et seulement celles-ci – qui peuvent être présentes dans le type chaîne contrainte.

NOTE – Par exemple, la définition suivante est soumise à une contrainte effective de taille:

A ::= IA5String (SIZE(1..4) | SIZE(10..15))

car on peut la réécrire sous forme d'une unique contrainte de taille qui s'applique à toutes les valeurs comme suit:

A ::= IA5String (SIZE(1..4 | 10..15))

tandis que l'expression suivante n'est soumise à aucune contrainte effective de taille car la chaîne peut avoir une longueur quelconque si elle ne contient pas d'autres caractères que 'a', 'b' et 'c':

B ::= IA5String (SIZE(1..4) | FROM("abc"))

3.8.9 contrainte effective d'alphabet permis (pour un type chaîne de caractères restreinte et contrainte): contrainte unique d'alphabet permis que l'on peut appliquer à un type natif chaîne de caractères à multiplicateur connu, dont l'effet sera de permettre tous les caractères – et seulement ceux-ci – qui peuvent occuper une position de caractère quelconque dans n'importe quelle valeur contenue dans le type chaîne contrainte.

NOTE – Une contrainte effective d'alphabet permis sera soit l'alphabet entier du type chaîne de caractères non restreinte ou une spécification d'alphabet permis qui se trouvera être un surensemble de toutes les contraintes d'alphabet permis qui sont imposées à ce type. Par exemple, dans la définition suivante:

Ax ::= IA5String (FROM("AB") | FROM("CD"))

Bx ::= IA5String (SIZE(1..4) | FROM("abc"))

où la chaîne "Ax" obéit à une contrainte effective d'alphabet permis qui consiste en l'alphabet IA5String, puisque aucune contrainte d'alphabet permis ne s'applique à toutes les valeurs de "Ax". Il en est de même pour "Bx". Par ailleurs, la définition suivante obéit à la contrainte effective d'alphabet permis pour les caractères "ABCDE" car elle spécifie une contrainte d'alphabet permis applicable à toutes ces valeurs:

A ::= IA5String (FROM("AB") | FROM("CD") | FROM("ABCDE"))

3.8.10 index d'énumération: nombre entier non négatif associé à un item dans un type énuméré (*enumerated*). Les index d'énumération sont déterminés en classant les items par ordre croissant de la valeur énumérée, puis en affectant un index d'énumération égal à 0 pour le premier item, à 1 pour le deuxième, etc., jusqu'au dernier élément de la liste ainsi ordonnée.

NOTE – Les items de base ("RootEnumeration") et les items additionnels ("AdditionalEnumeration") sont triés séparément.

3.8.11 extensibilité au codage compact: propriété d'un type dont la définition contient un marqueur d'extension qui affecte le codage selon les règles PER.

3.8.12 liste de champs: ensemble ordonné de valeurs de champ binaire et/ou de champ binaire calé à l'octet qui résulte de l'application des présentes règles de codage aux composantes d'une valeur.

NOTE – (Didactique) Le modèle employé dans la présente Recommandation | Norme internationale utilise le terme "liste de champs" pour indiquer une liste liée de registres contenant chacun une séquence codée, une longueur en bits ou un indicateur de calage à l'octet du "champ binaire" ou du "champ binaire calé à l'octet". Chaque séquence codée correspond à une valeur d'un type ASN.1. L'indicateur de calage à l'octet précise si cette séquence codée doit être calée sur une limite d'octet lorsqu'elle est utilisée pour former le codage complet de la valeur en syntaxe abstraite; il peut également préciser s'il convient d'insérer cette séquence codée immédiatement après le dernier bit de la précédente séquence codée du codage complet. La notion de "liste de champs" n'a de fonction que descriptive et ne propose aucune méthode de mise en œuvre.

3.8.13 longueur non définie: codage dont la longueur est supérieure à 64K – 1 ou dont la longueur maximale ne peut pas être déterminée d'après la notation ASN.1.

3.8.14 type de longueur fixe: type tel que l'on puisse déterminer – à partir de la notation de type (après application des seules contraintes visibles par les règles PER) – la valeur du déterminant de longueur le plus extérieur dans une séquence codée de ce type (au moyen des mécanismes spécifiés dans la présente Recommandation | Norme internationale) et tel que cette valeur soit la même pour toutes les valeurs possibles de ce type.

3.8.15 valeur fixe: valeur telle qu'elle puisse être déterminée (au moyen des mécanismes spécifiés dans la présente Recommandation | Norme internationale) comme étant la seule valeur permise (après application des seules contraintes visibles par les règles PER) du type dont elle dépend.

3.8.16 type chaîne de caractères à multiplicateur connu: type de chaîne de caractères restreinte dont le nombre d'octets codés est un multiple fixe et connu du nombre de caractères contenus dans la chaîne de caractères pour toutes les valeurs permises de la chaîne de caractères. Les types chaîne de caractères à multiplicateur connu sont les suivants: *IA5String, PrintableString, VisibleString, NumericString, UniversalString* et *BMPString*.

3.8.17 déterminant de longueur: compte (de bits, d'octets, de caractères ou de composantes) qui détermine la longueur de tout ou partie d'une séquence à codage PER.

3.8.18 nombre entier non négatif normalement petit: partie d'une séquence codée qui représente un entier non négatif et non délimité, mais dont les petites valeurs sont normalement plus fréquentes que les grandes.

3.8.19 longueur normalement petite: codage d'une longueur qui représente les valeurs d'une longueur non délimitée, mais telle que les petites valeurs de cette longueur soient normalement plus fréquentes que les grandes.

3.8.20 champ binaire calé à l'octet: produit issu d'une partie du mécanisme de codage, composé d'un ensemble ordonné d'éléments binaires qui n'est pas nécessairement un multiple de 8 mais qui doit commencer à une limite d'octet, dans le codage complet de la valeur de syntaxe abstraite.

3.8.21 codage d'entier binaire non négatif: codage d'un nombre entier contraint ou semi-contraint pour obtenir soit un champ binaire de longueur spécifiée, soit un champ binaire calé à l'octet et de longueur spécifiée, soit encore le nombre minimal d'octets permettant de représenter ce nombre entier sous la forme d'un entier binaire non négatif, dont le codage permet de représenter des nombres entiers supérieurs ou égaux à zéro, comme spécifié au 10.3.

NOTE – La valeur d'un nombre binaire en complément à deux est obtenue en numérotant les bits des octets qui le composent, en commençant par le bit 1 du dernier octet qui devient le bit 0 et en terminant par le bit 8 du premier octet. A chaque bit est affectée une valeur numérique de 2^N, N est la position du bit dans la séquence de numérotation précédente. La valeur du nombre binaire en complément à deux est obtenue en ajoutant les valeurs numériques affectées à chacun des bits qui sont mis à 1.

3.8.22 contrainte visible par les règles PER: instance d'utilisation de la notation de contraintes ASN.1 qui affecte le codage PER d'une valeur.

3.8.23 codage à compatibilité assurée: codage complet d'une valeur de syntaxe abstraite qui peut être décodée (y compris tous modules encastrés) sans connaissance de l'ensemble contextuel qui a été défini pour la couche présentation afin de former l'environnement d'exécution du codage.

3.8.24 nombre entier semi-contraint: nombre entier obéissant à des contraintes visibles par les règles PER de façon à être égal ou supérieur à une certaine valeur "lb", celle-ci étant une valeur permise et non pas un nombre entier contraint.

NOTE – Des nombres entiers semi-contraints apparaissent dans le codage de la longueur des types chaîne de caractères, chaîne d'octets et chaîne binaire non contraints (et parfois contraints), dans le compte du nombre de composantes contenues dans des types séquence-de et ensemble-de non contraints (et parfois contraints) et dans la valeur d'un type entier qui a été contraint à dépasser une certaine valeur minimale.

3.8.25 type simple: type qui n'est pas composite.

3.8.26 contextuellement dépendante: terme utilisé pour qualifier le cas où, si un certain nom de référence est utilisé pour évaluer un ensemble d'éléments, la valeur de celui-ci est considérée comme dépendant de ce nom de référence, que l'opération arithmétique en cours d'exécution réelle soit ou non telle que la valeur de l'ensemble d'éléments soit indépendante de la valeur d'ensemble d'éléments réellement affectée au nom de référence.

NOTE – Par exemple, la définition suivante de la variable métasyntaxique "Foo" dépend textuellement de la variable "Bar", bien que celle-ci n'ait aucun effet sur l'ensemble des valeurs de "Foo" (selon le 9.3.4, la contrainte sur la variable "Foo" n'est pas visible, puisque "Bar" est soumise à une contrainte tabulaire et que "Foo" dépend textuellement de "Bar").

```
MY-CLASS ::= CLASS { &name PrintableString, &age INTEGER } WITH SYNTAX{&name , &age}

MyObjectSet MY-CLASS ::= { {"Jack", 7} | {"Jill", 5} }

Bar ::= MY-CLASS.&age ({MyObjectSet})

Foo ::= INTEGER (Bar | 1..100)
```

3.8.27 nombre entier non contraint: nombre entier qui n'est pas soumis à des contraintes visibles par les règles PER.

NOTE – Des nombres entiers non contraints n'apparaissent que dans le codage d'une valeur de type entier.

4 Abréviations

ASN.1 Notation de syntaxe abstraite numéro un (*abstract syntax notation one*)

BER Règles de codage de base de l'ASN.1 (*basic encoding rules of ASN.1*)

PER Règles de codage compact de l'ASN.1 (*packed encoding rules of ASN.1*)

CER Règles de codage canonique de l'ASN.1 (*canonical encoding rules of ASN.1*)

16K 16384

32K 32768

48K 49152

64K 65536

5 Notation

La présente Recommandation | Norme internationale fait référence à la notation définie par la Rec. UIT-T X.680 | ISO/CEI 8824-1.

6 Conventions

6.1 La présente Recommandation | Norme internationale définit la valeur de chaque octet codé en utilisant les expressions "bit le plus significatif" et "bit le moins significatif".

NOTE – Les spécifications relatives aux couches inférieures utilisent la même notation pour définir l'ordre de transmission des bits sur un circuit série, ou d'affectation des bits à des canaux parallèles.

6.2 Pour les besoins de la présente Recommandation | Norme internationale, les bits d'un octet sont numérotés de 8 à 1, le 8^{e} bit étant le "bit le plus significatif" et le 1^{er} bit le "bit le moins significatif".

6.3 Dans la présente Recommandation | Norme internationale, le terme "octet" a souvent le sens de "huit éléments binaires". L'emploi du terme "octet" au lieu de "8 éléments binaires" n'implique aucune prescription d'alignement. Si celui-ci est recherché, cela est explicitement déclaré dans la présente Recommandation | Norme internationale.

7 Règles de codage définies dans la présente Recommandation | Norme internationale

7.1 La présente Recommandation | Norme internationale spécifie quatre règles de codage (ainsi que leurs identificateurs d'objet associés). Ces quatre règles pourront être utilisées pour coder et décoder les valeurs d'une syntaxe abstraite définie comme contenant les valeurs d'un seul type ASN.1 (connu). Le présent article décrit l'applicabilité et les propriétés de ces règles.

7.2 Si l'on ne connaît pas le type de la valeur à coder, il n'est pas possible de déterminer la structure du codage (selon l'un des algorithmes des règles de codage compact). En particulier, la fin d'une séquence codée ne peut pas être déterminée d'après cette séquence si l'on ne connaît pas le type qui est codé.

7.3 Les codages PER sont toujours à compatibilité assurée, à condition que les valeurs abstraites des types EXTERNAL, EMBEDDED PDV et CHARACTER STRING soient contraintes de manière à empêcher l'acheminement d'identificateurs de contexte de présentation.

7.4 L'algorithme de règle de codage le plus général, spécifié dans la présente Recommandation | Norme internationale, est de type BASIC-PER, qui ne produit généralement pas de codage canonique.

7.5 Un deuxième algorithme de règle de codage, spécifié dans la présente Recommandation | Norme internationale, est de type CANONICAL-PER, qui produit généralement des codages canoniques. Il est défini sous la forme d'une restriction des choix dépendant de la mise en œuvre dans le codage de type BASIC-PER. L'algorithme CANONICAL-PER produit des codages canoniques qui ont des applications lorsqu'il faut appliquer des authentificateurs à des valeurs abstraites, comme décrit dans l'Annexe D de la Rec. UIT-T X.690 | ISO/CEI 8825-1.

NOTE – Toute mise en œuvre codable selon les règles de type CANONICAL-PER est conforme aux règles de codage BASIC-PER. Toute mise en œuvre décodable selon les règles de type BASIC-PER est conforme aux règles de décodage CANONICAL-PER. Les codages effectués selon les règles CANONICAL-PER sont donc autorisés par les règles BASIC-PER.

7.6 Si un type codé selon les règles BASIC-PER ou CANONICAL-PER contient des types comme EMBEDDED PDV, CHARACTER STRING ou EXTERNAL, le codage extérieur perd son assurance de compatibilité, à moins que la syntaxe de transfert utilisée pour tous ces types (EMBEDDED PDV, CHARACTER STRING et EXTERNAL) soit elle-même à compatibilité assurée. Si un type codé selon les règles BASIC-PER ou CANONICAL-PER contient des types comme EMBEDDED PDV, EXTERNAL ou CHARACTER STRING, le codage extérieur perd son caractère canonique, à moins que la syntaxe de transfert utilisée pour tous ces types (EMBEDDED PDV, EXTERNAL et CHARACTER STRING) ne soit elle-même canonique.

NOTE – Les syntaxes de transfert de caractères, prenant en charge toutes les syntaxes abstraites en mode caractère de la forme {iso standard 10646 level-1 (1) ....} sont canoniques. Celles qui prennent en charge des syntaxes de la forme {iso standard 10646 level-2 (2) ....} et {iso standard 10646 level-3 (3) ....} ne sont pas toujours canoniques. Toutes les syntaxes de transfert de caractères susmentionnées sont à compatibilité assurée.

7.7 Les règles BASIC-PER et CANONICAL-PER ont chacune deux variantes: ALIGNED et UNALIGNED. Dans la variante ALIGNED, des bits de bourrage sont insérés de temps en temps afin de restaurer l'alignement en octets. Dans la variante UNALIGNED, aucun bit de bourrage n'est jamais inséré.

7.8 Il n'existe aucune possibilité d'interfonctionnement entre la variante ALIGNED et la variante UNALIGNED.

7.9 Les codages compacts (selon les règles PER) ne sont autodélimitants que si l'on connaît le type de la valeur codée. Les codages sont toujours un multiple de 8 éléments binaires. Lorsqu'ils sont acheminés dans un type EXTERNAL, ils doivent figurer dans l'option OCTET STRING, à moins que le type EXTERNAL soit lui-même en codage compact, auquel cas la valeur peut être codée sous la forme d'un unique type ASN.1 (c'est-à-dire comme un type ouvert). Lorsque les codages compacts sont acheminés dans un protocole de couche présentation de l'OSI, le "codage complet" (tel que défini dans la Rec. UIT-T X.226 | ISO/CEI 8823-1) doit être utilisé avec l'option OCTET STRING.

7.10 Les règles de la présente Recommandation | Norme internationale s'appliquent aux deux algorithmes et aux deux variantes, sauf indication contraire.

7.11 L'Annexe C est informative et donne des recommandations sur les combinaisons de règles PER à mettre en œuvre afin de maximiser les chances d'interfonctionnement.

8 Conformité

8.1 La conformité dynamique est spécifiée à partir de l'article 9.

8.2 La conformité statique est spécifiée par les règles d'application des présentes règles de codage compact.

NOTE – L'Annexe C de la présente Recommandation | Norme internationale donne des directives sur la conformité statique afin d'assurer le support des deux variantes des deux algorithmes de codage. Ces directives sont conçues de façon à assurer l'interfonctionnement, tout en admettant que, pour certaines applications, il peut être préférable de suivre des règles de codage qui ne sont ni à compatibilité assurée ni canoniques.

8.3 Les règles contenues dans la présente Recommandation | Norme internationale sont spécifiées en termes de procédure de codage. Les mises en œuvre ne sont pas tenues de refléter intégralement la procédure spécifiée, à condition que la chaîne binaire produite comme codage complet d'une valeur de syntaxe abstraite, soit identique à l'une des chaînes binaires spécifiées dans la présente Recommandation | Norme internationale pour la syntaxe de transfert applicable.

8.4 Les mises en œuvre effectuant le décodage sont tenues de produire la valeur de syntaxe abstraite correspondant à toute chaîne binaire reçue en provenance d'un expéditeur se conformant aux règles de codage indiquées dans la syntaxe de transfert associée aux données à décoder.

NOTES

1 En général, on ne définit pas de variantes de codage pour les règles BASIC-PER qui sont explicitement déclarées dans la présente Recommandation | Norme internationale. Le codage BASIC-PER devient canonique lorsque l'on spécifie un fonctionnement à compatibilité assurée et que l'on restreint certaines des options de codage indiquées par d'autres Normes ISO/CEI citées en référence. L'algorithme CANONICAL-PER offre une variante, aussi bien aux règles de codage distinctif (DER) qu'aux règles de codage canonique (CER) (voir la Rec. UIT-T X.690 | ISO/CEI 8825-1), lorsqu'il est nécessaire de disposer d'un codage canonique à compatibilité assurée.

2 Lorsque l'algorithme CANONICAL-PER est utilisé pour produire un codage canonique, il est recommandé que toute valeur chiffrée à codage dispersé qui en est dérivée dispose d'un identificateur d'algorithme associé qui indique que l'algorithme CANONICAL-PER a été utilisé pour transformer la valeur abstraite en une chaîne binaire initiale (dispersée par la suite).

9 Méthode de codage utilisée pour les règles PER

9.1 Utilisation de la notation de types

9.1.1 Les présentes règles de codage font spécifiquement appel à la notation des types ASN.1 qui est spécifiée dans la Rec. UIT-T X.680 | ISO/CEI 8824-1. Elles ne peuvent être appliquées que pour coder les valeurs d'un seul type ASN.1, spécifié au moyen de cette notation.

9.1.2 Ces règles dépendent en particulier, mais non exclusivement, de la conservation des informations suivantes dans le modèle de type et de valeur ASN.1 sur lequel est fondée l'utilisation de cette notation:

a) l'imbrication de types choice à l'intérieur de types choice;

b) les étiquettes attribuées aux composantes d'un type set et aux variantes d'un type choice ainsi que les valeurs attribuées à une énumération;

c) le fait qu'une composante d'un type set ou sequence soit ou non facultative;

d) le fait qu'une composante d'un type set ou sequence possède ou non une valeur par défaut;

e) la restriction de l'étendue des valeurs d'un type en raison de l'application de contraintes visibles (seulement) par les règles PER;

f) le fait qu'une composante soit de type ouvert;

g) le fait qu'un marqueur d'extension soit présent.

9.2 Utilisation d'étiquettes pour établir un ordre canonique

La présente Recommandation | Norme internationale prescrit que les composantes d'un type set ou choice soient en relation d'ordre canonique, indépendamment de leur ordre contextuel. L'ordre canonique est déterminé par tri des étiquettes attribuées aux composantes, comme spécifié au 6.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

9.3 Contraintes visibles par les règles PER

NOTE – Le fait que certaines contraintes de notation ASN.1 puissent ne pas être visibles par les règles PER dans le cadre d'un codage et d'un décodage n'a aucune incidence que ce soit sur l'utilisation de telles contraintes pour traiter des erreurs détectées au cours du décodage; il n'en découle pas non plus que les valeurs violant de telles contraintes puissent être émises par un expéditeur conforme.

9.3.1 Les contraintes exprimées sous forme de texte en clair ou de commentaire ASN.1 ne sont pas visibles par les règles PER.

9.3.2 Les contraintes variables ne sont pas visibles par les règles PER (voir 10.4 et 10.5 de la Rec. UIT-T X.683 | ISO/CEI 8824-4).

9.3.3 Les contraintes tabulaires ne sont pas visibles par les règles PER (voir la Rec. UIT-T X.682 | ISO/CEI 8824-3).

9.3.4 Les contraintes dont l'évaluation est contextuellement dépendante d'une contrainte tabulaire ou d'une contrainte relationnelle sur composantes ne sont pas visibles par les règles PER (voir la Rec. UIT-T X.682 | ISO/CEI 8824-3).

9.3.5 Les contraintes relationnelles sur composantes ne sont pas visibles par les règles PER (voir la Rec. UIT-T X.682 | ISO/CEI 8824-3).

9.3.6 Les contraintes sur des types chaîne de caractères restreinte autres que des types chaîne de caractères à multiplicateur connu (voir article 34 de la Rec. UIT-T X.680 | ISO/CEI 8824-1) ne sont pas visibles par les règles PER (voir 3.8.16).

9.3.7 Sous réserve de ce qui précède, toutes les contraintes de taille sont visibles par les règles PER.

9.3.8 La contrainte effective de taille pour un type contraint est une contrainte de taille unique telle qu'une taille ne soit permise que si ce type contraint possède une valeur qui a cette taille (permise). Si le type contraint possède des valeurs de taille qui n'obéissent pas à cette contrainte, il n'y a pas de contrainte effective de taille.

9.3.9 Les contraintes d'alphabet permis sur des types chaîne de caractères à multiplicateur connu sont visibles par les règles PER.

9.3.10 La contrainte effective d'alphabet permis sur un type contraint est une contrainte sur alphabet permis isolée, telle qu'un caractère ne soit permis que si ce type contraint possède une valeur qui contient ce caractère. Si tous les caractères du type soumis à la contrainte peuvent être présents dans une valeur du type contraint, la contrainte effective d'alphabet permis est l'ensemble des caractères définis pour le type non contraint.

NOTES

1 Dans la définition d'un type contraint, plusieurs contraintes visibles par les règles PER peuvent être appliquées, directement ou au moyen de sous-types confinés (*ContainedSubtype*).

2 Voir Annexe B, pour les observations sur l'effet de la combinaison de contraintes individuellement visibles par les règles PER.

9.3.11 Une contrainte de type interne, appliquée à un type REAL, est visible par les règles PER.

9.3.12 Une contrainte de type interne, appliquée à un type chaîne de caractères non restreinte ou à un type valeur PDV encapsulée, n'est visible par les règles PER que lorsqu'elle sert à restreindre à une seule variante la valeur de la composante "syntaxes", ou lorsqu'elle sert à restreindre la composante "identification" à la seule valeur "fixed" (voir articles 24 et 27).

9.3.13 Les contraintes sur les types utiles ne sont pas visibles par les règles PER.

9.3.14 Sous réserve de ce qui précède, toutes les autres contraintes ne sont visibles par les règles PER que si elles sont appliquées à un type integer ou, si l'on exclut les contraintes monovaluantes, appliquées à un type chaîne de caractères à multiplicateur connu.

9.3.15 Si une contrainte visible par les règles PER, autre que d'alphabet permis, possède un marqueur d'extension, le type auquel elle s'applique est défini comme étant extensible au codage compact.

NOTES

1 Si un marqueur d'extension est présent dans un type *ConstraintSpec* qui n'est pas visible par les règles PER et si aucun autre marqueur d'extension n'est présent dans la contrainte, ce type est codé par les règles PER comme s'il ne possédait aucun marqueur d'extension.

2 Si plusieurs spécifications de contrainte de taille (*SizeConstraint*) sont appliquées à un type et que l'une d'elles soit extensible, le type est codé par les règles PER comme si le marqueur d'extension était présent dans toutes les spécifications de contrainte de taille.

9.3.16 Un type est également extensible au codage compact si une des conditions suivantes est vérifiée:

a) ce type est dérivé d'un type ENUMERATED (par sous-typage, par référenciation de type ou par étiquetage) et la production "Enumerations" contient un marqueur d'extension;

b) ou bien ce type est dérivé d'un type SEQUENCE (par sous-typage, par référenciation de type ou par étiquetage) et la production "ComponentTypeLists" ou "SequenceType" contient un marqueur d'extension;

c) ou bien ce type est dérivé d'un type SET (par sous-typage, par référenciation de type ou par étiquetage) et la production "ComponentTypeLists" ou "SetType" contient un marqueur d'extension;

d) ou bien ce type est dérivé d'un type CHOICE (par sous-typage, par référenciation de type ou par étiquetage) et la production "AlternativeTypeLists" contient un marqueur d'extension.

9.4 Modèle utilisé pour coder les types et les valeurs

9.4.1 En notation ASN.1, un type est simple ou construit au moyen d'autres types. Cette notation permet de faire appel à la référenciation et à l'étiquetage des types. Dans le cadre des présentes règles de codage, la référenciation et l'étiquetage des types n'ont aucune incidence sur le codage et sont invisibles dans le modèle, sauf ce qui est indiqué au 9.2. La notation permet également d'appliquer des contraintes et de spécifier des erreurs. Les contraintes visibles par les règles PER sont présentes dans le modèle en tant que restrictions sur les valeurs d'un type. D'autres contraintes et spécifications d'erreur n'ont pas d'incidence sur le codage et sont invisibles dans le modèle type et valeur PER.

9.4.2 Une valeur à coder peut être considérée comme étant simple ou composite, c'est-à-dire construite au moyen des mécanismes de structuration à partir de composantes qui sont des valeurs simples ou des valeurs composites, ce qui correspond à la structure des définitions de type en notation ASN.1.

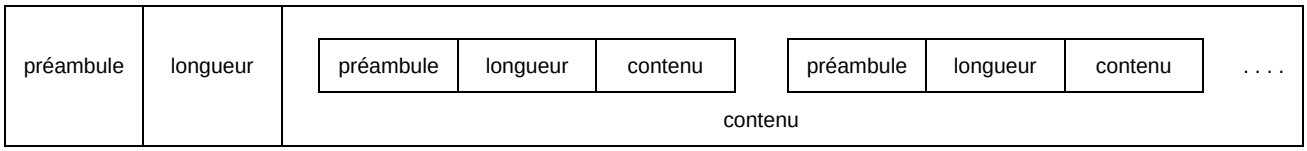
9.5 Structure d'une expression codée

9.5.1 Les présentes règles de codage spécifient:

a) le codage d'une valeur simple pour la transformer en liste de champs;

b) le codage d'une valeur composite pour la transformer en liste de champs, au moyen des listes de champs produites par l'application des présentes règles de codage aux composantes de la valeur composite;

c) la transformation de la liste des champs de la valeur la plus externe en codage complet de la valeur de la syntaxe abstraite (voir 10.1).



NOTE – Le préambule, la longueur et le contenu sont des "champs" dont la concaténation forme une "liste de champs". La liste de champs d'un type composite différent du type choix (*Choice*) peut être formée par la concaténation de champs de valeur différente. Le préambule, la longueur et/ou le contenu peuvent ne pas figurer, quelle que soit leur valeur.

Figure 1 – Codage d'une valeur composite dans une liste de champs

9.5.2 Le codage d'une composante de valeur de données est constitué d'une des deux manières suivantes:

a) comprend trois parties, comme représenté à la Figure 1, qui apparaissent dans l'ordre suivant:

1) un préambule (voir articles 18, 20 et 22);

2) un déterminant de longueur (voir 10.9);

3) un contenu;

b) ou bien (si le contenu est important), un nombre quelconque de parties (comme représenté à la Figure 2), dont la première est un préambule (voir articles 18, 20 et 22), les parties suivantes étant des paires de champs binaires calés à l'octet, le premier champ étant un déterminant de longueur pour un fragment du contenu et le deuxième étant ce fragment de contenu; la dernière paire de champs est identifiée par la partie contenant le déterminant de longueur, comme spécifié au 10.9.

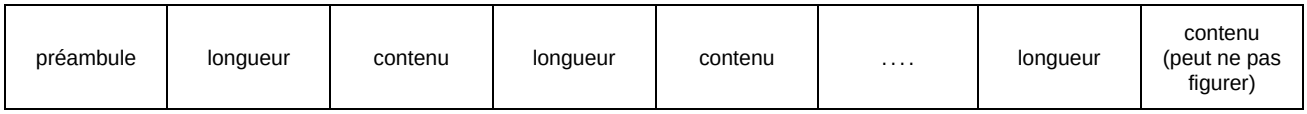


Figure 2 – Codage d'une valeur de données longue

9.5.3 Chacune des parties mentionnées au 9.5.2 produit un des résultats suivants:

a) un champ nul (néant);

b) un champ binaire;

c) un champ binaire calé à l'octet;

d) une liste de champs pouvant contenir des champs binaires, ou des champs binaires calés à l'octet, ou les deux sortes.

9.6 Types à coder

9.6.1 Les articles suivants spécifient le codage des types suivants pour créer une liste de champs: booléen, entier, énuméré, réel, chaîne binaire, chaîne d'octets, néant, séquence, séquence-de, ensemble, ensemble-de, choix, ouvert, identificateur d'objet, valeurs PDV encapsulées, externe, chaîne de caractères restreinte et chaîne de caractères non restreinte.

9.6.2 Le type ANY, défini dans la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990 doit être codé comme un type ouvert.

9.6.3 Le type de sélection doit être codé sous forme de séquence codée du type sélectionné.

9.6.4 Le codage d'un type étiqueté n'est pas traité dans la présente Recommandation | Norme internationale car, sauf dans le cas stipulé au 9.2, l'étiquetage n'est pas visible dans le modèle type et valeur utilisé pour les présentes règles de codage. Un type étiqueté est donc codé comme le type correspondant qui a été étiqueté.

9.6.5 Les "types utiles" suivants, définis à l'article 38 de la Rec. UIT-T X.680 | ISO/CEI 8824-1, doivent être codés comme s'ils avaient été remplacés par leurs définitions selon la Rec. UIT-T X.680 | ISO/CEI 8824-1:

- temps généralisé;
- temps universel;
- descripteur d'objet.

Les contraintes sur les types utiles ne sont pas visibles par les règles PER.

10 Procédures de codage

10.1 Production du codage complet

10.1.1 La liste de champs produite par l'application des règles de la présente Recommandation | Norme internationale à la valeur la plus externe doit être utilisée pour produire le codage complet de la valeur en syntaxe abstraite, comme suit: on prend tour à tour chaque champ de la liste et on le concatène à la fin de la chaîne binaire qui forme le codage complet de la valeur en syntaxe abstraite, tout en ajoutant des bits 0 de bourrage, comme spécifié ci-après.

10.1.2 Dans la variante UNALIGNED des présentes règles de codage, tous les champs doivent être concaténés sans bourrage. Si le résultat du codage de la valeur la plus externe est une chaîne binaire vide, celle-ci doit être remplacée par un octet unique dont tous les bits sont forcés à 0. Si le résultat est une chaîne binaire non vide et que cette chaîne ne soit pas un multiple de huit éléments binaires (zéro à sept), les bits 0 doivent être ajoutés à cette chaîne afin d'obtenir un multiple de huit éléments binaires.

10.1.3 Dans la variante ALIGNED des présentes règles de codage, tous les champs binaires de la liste de champs doivent être concaténés sans bourrage et tous les champs binaires alignés en octets doivent être concaténés après adjonction d'une série de (zéro à sept) bits 0 afin que la longueur du codage produit jusque-là soit un multiple de huit éléments binaires. Si le résultat du codage de la valeur la plus externe est une chaîne binaire vide, celle-ci doit être remplacée par un octet unique dont tous les bits sont forcés à 0. Si le résultat est une chaîne binaire non vide et que cette chaîne ne soit pas un multiple de huit éléments binaires (zéro à sept), les bits 0 doivent être ajoutés à cette chaîne afin d'obtenir un multiple de huit éléments binaires.

NOTE – Le codage de la valeur la plus extérieure est la chaîne binaire vide si, par exemple, la valeur de la syntaxe abstraite est du type néant ou un type entier contraint à être une valeur unique.

10.1.4 La chaîne binaire résultante est le codage complet de la valeur de syntaxe abstraite.

10.2 Champs de type ouvert

10.2.1 Pour coder un champ de type ouvert, la valeur du type occupant déjà ce champ doit être codée de manière à créer une liste de champs, laquelle est ensuite convertie dans la séquence de codage complète d'une valeur de syntaxe abstraite, comme spécifié au 10.1, afin de produire une chaîne d'octets de longueur "n" (par exemple).

10.2.2 La liste des champs de la valeur dans laquelle le type ouvert doit être imbriqué doit alors être complétée (comme spécifié au 10.9) d'une longueur non contrainte de "n" (en octets) et d'un champ binaire calé à l'octet, contenant les éléments binaires produits au 10.2.1.

NOTE – Lorsque le nombre d'octets contenus dans le type ouvert est grand, on applique les procédures de fragmentation indiquées au 10.9 et le codage du type ouvert sera haché sans tenir compte de la position des limites de fragment dans le codage du type occupant le champ de type ouvert.

10.3 Codage sous forme d'un entier binaire non négatif

NOTE – (Didactique) Ce paragraphe précise le sens du terme "codage d'entier binaire non négatif", ce nombre étant placé dans un champ dont le nombre d'éléments binaires est fixe ou dans un champ dont le nombre d'octets est le minimum requis pour contenir ce nombre.

10.3.1 Les paragraphes suivants se rapportent à la production d'un codage entier binaire non négatif représentant un nombre entier non négatif afin d'obtenir un champ binaire de longueur spécifiée, un seul octet, deux octets ou le nombre minimal d'octets permettant de représenter la valeur. Ce paragraphe spécifie le codage précis à appliquer lorsque de telles références sont faites.

10.3.2 Le bit initial du champ est défini comme étant le bit le plus significatif du premier octet et le bit final du champ est défini comme étant le bit le moins significatif du dernier octet.

10.3.3 Pour la définition suivante seulement, les bits doivent être numérotés comme suit: zéro pour le bit final du champ, un pour le bit suivant et ainsi de suite jusqu'au bit initial du champ.

10.3.4 Dans un codage d'entier binaire non négatif, la valeur du nombre entier représenté par le code doit être la somme des valeurs spécifiées pour chaque bit. Un bit qui est mis à "0" a la valeur zéro. Un bit numéro "n" qui est mis à "1" a la valeur 2^n.

10.3.5 La séquence codée dont la somme (comme définie ci-dessus) atteint la valeur en cours de codage est une représentation codée de cette valeur.

NOTE – Lorsque la taille du champ codé est fixe (champ binaire de longueur spécifiée, octet unique, ou double octet), il existe un unique codage dont la somme atteint la valeur à coder.

10.3.6 Un codage d'entier binaire non négatif sur le nombre minimal d'octets pour représenter ce nombre entier (ce qui ne préjuge pas le nombre d'octets à utiliser pour le codage) possède un champ qui est un multiple de huit bits et qui satisfait la condition que les huit bits initiaux du champ n'ont pas la valeur zéro, sauf si le champ a exactement une longueur de huit bits.

NOTE – Cette condition est nécessaire et suffisante pour produire un codage unique.

10.4 Codage sous forme d'un entier binaire en complément à deux

NOTE – (Didactique) Ce paragraphe précise le sens du terme "codage d'entier binaire en complément à deux", par lequel un entier signé est inséré dans un champ ayant le nombre minimal d'octets pour le contenir. Ces procédures feront l'objet de références dans des spécifications de codage ultérieures.

10.4.1 Les paragraphes suivants se rapportent à la production d'un codage d'entier binaire en complément à deux représentant un nombre entier (qui peut être négatif, nul ou positif) afin d'obtenir le nombre minimal d'octets pour représenter la valeur. Ce paragraphe spécifie le codage précis à appliquer lorsque de telles références sont faites.

10.4.2 Le bit initial du champ est défini comme étant le bit le plus significatif du premier octet et le bit final du champ est défini comme étant le bit le moins significatif du dernier octet.

10.4.3 Pour la définition suivante seulement, les bits doivent être numérotés comme suit: zéro pour le bit final du champ, un pour le bit suivant et ainsi de suite jusqu'au bit initial du champ.

10.4.4 Dans un codage d'entier binaire en complément à deux, la valeur du nombre entier représenté par le code doit être la somme des valeurs spécifiées pour chaque bit. Un bit qui est mis à zéro a la valeur "0". Un bit numéro "n" qui est mis à "1" a la valeur 2^n, sauf le bit initial qui prend alors la valeur (négative) $–2^n$.

10.4.5 Toute séquence codée dont la somme (comme définie ci-dessus) atteint la valeur en cours de codage est une représentation codée de cette valeur.

10.4.6 Un codage d'entier binaire en complément à deux sur le nombre minimal d'octets pour représenter ce nombre entier possède un champ dont la longueur est un multiple de huit éléments binaires et qui satisfait la condition que les neuf éléments binaires initiaux du champ ne doivent pas avoir tous la valeur zéro et ne doivent pas tous avoir la valeur un.

NOTE – Cette condition est nécessaire et suffisante pour produire un codage unique.

10.5 Codage d'un nombre entier contraint

NOTE – (Didactique) Ce paragraphe est cité en référence par d'autres paragraphes et fait lui-même référence à des paragraphes antérieurs concernant la production d'un codage d'entier binaire positif ou en complément à deux. Pour la variante UNALIGNED, la valeur est toujours codée sur le nombre minimal de bits nécessaire pour représenter la plage (définie au 10.5.3). Le reste de la présente note traite de la variante ALIGNED. Lorsque la plage est inférieure ou égale à 255, la valeur code pour un champ binaire de longueur minimale dans la plage. Lorsqu'elle est exactement égale à 256, la valeur code pour un champ binaire calé à l'octet, d'une longueur d'un octet. Lorsque la plage est comprise entre 257 et 64K, la valeur code pour un champ binaire calé à l'octet, d'une longueur de deux octets. Lorsque la plage est plus grande que 64K, elle est ignorée et la valeur code pour un champ binaire calé à l'octet dont le nombre d'octets est le minimum pour représenter la valeur. Dans ce dernier cas, des procédures exposées plus loin (voir 10.9) codent également un champ indicateur de longueur du codage (habituellement sur un seul octet). Dans les cas précédents, la longueur du codage est indépendante de la valeur à coder et cette longueur n'est pas explicitement codée.

10.5.1 Ce paragraphe spécifie une transformation d'un nombre entier contraint en un champ binaire ou en un champ binaire calé à l'octet. Il est cité en référence par des paragraphes ultérieurs de la présente Recommandation | Norme internationale.

10.5.2 Les procédures exposées dans ce paragraphe ne sont exécutées que si un nombre entier contraint est à coder et si les valeurs des bornes inférieure ("lb") et supérieure ("ub") ont été déterminées sur la base de la notation de type (après application des contraintes visibles par les règles PER).

NOTE – Il n'est pas possible de déterminer la borne inférieure si MIN prend une valeur infinie, ni la borne supérieure si MAX prend une valeur infinie. Ainsi, aucune des deux bornes ne peut être déterminée dans la déclaration INTEGER(MIN..MAX).

10.5.3 La grandeur "plage" est par exemple définie comme ayant la valeur de l'entier ("ub" – "lb" + 1) et la valeur à coder est "n".

10.5.4 Si "plage" a la valeur 1, le résultat du codage doit être un champ binaire vide (aucun élément binaire).

10.5.5 Cinq autres cas (donnant des codages différents) sont à considérer, dont un s'applique à la variante UNALIGNED et quatre à la variante ALIGNED.

10.5.6 Dans le cas de la variante UNALIGNED, la valeur ("n" – "lb") doit être codée comme un entier binaire non négatif, dans un champ binaire, comme spécifié au 10.3, avec le nombre minimal de bits nécessaire pour représenter la plage.

NOTE – Si la "plage" satisfait à l'inégalité suivante: 2^m < "plage" <= 2^{m+1}, le nombre de bits sera égal à m + 1.

10.5.7 Dans le cas de la variante ALIGNED, le codage dépend de la plage:

a) "plage" inférieure ou égale à 255 (cas du champ binaire);

b) "plage" exactement égale à 256 (cas de l'octet unique);

c) "plage" supérieure à 256 et inférieure ou égale à 64K (cas des deux octets);

d) "plage" supérieure à 64K (cas d'une longueur non définie).

10.5.7.1 (Cas du champ binaire) Si la "plage" est inférieure ou égale à 255, l'application de ce paragraphe nécessite la production d'un champ binaire dont le nombre de bits est conforme au tableau ci-dessous et dont la valeur ("n" – "lb") est un entier binaire non négatif codé dans un champ binaire comme spécifié au 10.3.

"Plage"	Taille du champ binaire (en bits)
2	1
3, 4	2
5, 6, 7, 8	3
9 à 16	4
17 à 32	5
33 à 64	6
65 à 128	7
129 à 255	8

10.5.7.2 (Cas de l'octet unique) Si la "plage" a la valeur 256, la valeur de l'entier ("n" – "lb") doit être codée sur un octet dans un champ binaire calé à l'octet, sous forme d'un entier binaire non négatif codé comme spécifié au 10.3.

10.5.7.3 (Cas des deux octets) Si la "plage" a une valeur supérieure ou égale à 257 et inférieure ou égale à 64K, la valeur ("n" – "lb") doit être codée sur deux octets dans un champ binaire calé à l'octet, sous forme d'un entier binaire non négatif codé comme spécifié au 10.3.

10.5.7.4 (Cas de la longueur non définie) Sinon, la valeur ("n" – "lb") doit être codée sous forme d'un entier binaire non négatif dans un champ binaire calé à l'octet, sur le nombre minimal d'octets comme spécifié au 10.3. Ce nombre d'octets ("len") de codage sera utilisé par d'autres paragraphes faisant référence au présent paragraphe pour spécifier un codage de longueur.

10.6 Codage d'un nombre entier non négatif normalement petit

NOTE – (Didactique) Cette procédure est utilisée lors du codage d'un nombre entier non négatif que l'on estime petit mais dont la taille peut être illimitée à cause de la présence d'un marqueur d'extension. Ce sera par exemple un index de choix.

10.6.1 Si le nombre entier non négatif "n" est inférieur ou égal à 63, un champ binaire contenant un seul bit (mis à 0) doit être ajouté à la liste des champs et le nombre "n" doit être codé comme un entier binaire non négatif représenté sous la forme d'un champ binaire de 6 bits.

10.6.2 Si le nombre "n" est supérieur ou égal à 64, un champ binaire contenant un seul bit (mis à 1) doit être ajouté à la liste des champs. Le nombre "n" doit ensuite être codé comme un nombre entier semi-contraint, la limite "lb" étant égale à 0, et les procédures du 10.9 étant appliquées pour annexer ce champ à la liste des champs, précédé d'un déterminant de longueur.

10.7 Codage d'un nombre entier semi-contraint

NOTE – (Didactique) Cette procédure est utilisée lorsqu'on peut identifier une limite inférieure mais pas de limite supérieure. Cette procédure de codage transfère le décalage par rapport à la limite inférieure sous la forme d'un entier binaire non négatif sur le nombre minimal d'octets. Elle nécessite un codage explicite de longueur (normalement sur un seul octet) comme spécifié dans des procédures qui seront exposées plus bas.

10.7.1 Ce paragraphe spécifie l'application d'un nombre entier semi-contraint sur un champ binaire calé à l'octet. Il est invoqué par des paragraphes ultérieurs de la présente Recommandation | Norme internationale.

10.7.2 Les procédures de ce paragraphe ne sont invoquées que si un nombre entier semi-contraint (par exemple "n") est à coder et si la valeur de la limite "lb" a été déterminée d'après la notation du type (après application des contraintes visibles par les règles PER).

NOTE – Il n'est pas possible de déterminer la borne inférieure si MIN prend une valeur infinie. Ainsi, il n'est pas possible de déterminer la borne inférieure dans l'expression INTEGER(MIN..MAX).

10.7.3 Les procédures de ce paragraphe produisent toujours le cas de la longueur non définie.

10.7.4 (Cas de la longueur non définie) La valeur ("n" – "lb") doit être codée sous la forme d'un entier binaire non négatif dans un champ binaire calé à l'octet, avec le nombre minimal d'octets spécifié au 10.3; le nombre d'octets ("len") utilisé pour le codage sera repris par d'autres paragraphes faisant référence à ce paragraphe pour spécifier un codage de longueur.

10.8 Codage d'un nombre entier non contraint

NOTE – (Didactique) Ce cas ne se produit que lors du codage de la valeur d'un type entier sans limite inférieure. La procédure code la valeur sous la forme d'un entier binaire en complément à deux, sur le nombre minimal d'octets nécessaire pour contenir le codage. Cette procédure nécessite un codage explicite de longueur (normalement sur un seul octet) comme spécifié dans des procédures exposées ultérieurement.

10.8.1 Ce paragraphe spécifie le mappage d'un nombre entier non contraint (par exemple "n") sur un champ binaire calé à l'octet. Il est appliqué par des paragraphes ultérieurs de la présente Recommandation | Norme internationale.

10.8.2 Les procédures de ce paragraphe produisent toujours le cas de la longueur non définie.

10.8.3 (Cas de la longueur non définie) La valeur "n" doit être codée sous la forme d'un entier binaire en complément à deux dans un champ binaire aligné en octets, avec le nombre minimal d'octets spécifié au 10.4; le nombre d'octets ("len") utilisé pour le codage sera repris par d'autres paragraphes faisant référence à ce paragraphe pour spécifier un codage de longueur.

10.9 Règles générales pour le codage d'un déterminant de longueur

NOTES

1 (Didactique) Les procédures de ce paragraphe sont appliquées lorsqu'un champ de longueur explicite est requis pour une partie du codage, que la valeur mesurée de cette longueur ait ou non une limite supérieure (par contraintes visibles par les règles PER). La partie du codage à laquelle cette longueur s'applique peut être une chaîne binaire (de longueur mesurée en bits), une chaîne d'octets (de longueur mesurée en octets), une chaîne de caractères à multiplicateur connu (de longueur mesurée en caractères), ou une liste de champs (de longueur mesurée en nombre de composantes d'un type séquence-de ou ensemble-de).

2 (Didactique) Dans le cas de la variante ALIGNED, si la valeur mesurée de la longueur possède une limite supérieure qui est inférieure à 64K, cette longueur est codée par nombre entier contraint. Pour des plages suffisamment petites, il en résulte un champ binaire, sinon, la longueur non contrainte (par exemple de valeur "n") sera codée dans un champ binaire calé à l'octet selon l'une des trois méthodes suivantes (par ordre croissant des tailles):

a) (longueur "n" inférieure à 128) un seul octet contenant "n" avec le bit 8 mis à zéro;

b) (longueur "n" inférieure à 16K) deux octets contenant "n" avec le bit 8 du premier octet mis à 1 et le bit 7 mis à 0;

c) (grande longueur "n") un seul octet contenant une valeur "m" avec le bit 8 mis à 1 et le bit 7 mis à 1. La valeur "m" est comprise entre 1 et 4 et cette longueur indique qu'un fragment des données doit suivre (soit un multiple "m" de 16K items). Pour toutes les valeurs de "m", le fragment est alors suivi d'un autre codage de longueur pour le reste des données.

3 (Didactique) Dans la variante UNALIGNED, si la longueur mesurée possède une limite supérieure qui est inférieure à 64K, le codage de nombre entier contraint est utilisé pour coder cette longueur sur le nombre minimal de bits nécessaire pour représenter la plage. Sinon, la longueur non contrainte (par exemple "n") est codée dans un champ binaire comme décrit dans la Note 2 ci-dessus.

10.9.1 Ce paragraphe n'est pas invoqué si, conformément à la spécification de paragraphes ultérieurs, la valeur du déterminant de longueur ("n") est contrainte par la définition du type (contraint par des contraintes visibles par les règles PER) à une valeur inférieure à 64K.

10.9.2 Ce paragraphe est invoqué afin de compléter la liste des champs d'un champ ou d'une liste de champs précédé(e) d'un déterminant de longueur "n" qui indique une des grandeurs suivantes:

a) la longueur en octets d'un champ associé (mesuré en octets);

b) la longueur en bits d'un champ associé (mesuré en bits);

c) le nombre de composantes codées dans une liste associée de champs (mesuré en composantes d'un type ensemble-de ou séquence-de);

d) le nombre de caractères de la valeur d'un type associé chaîne de caractères à multiplicateur connu (mesuré en caractères).

10.9.3 (Variante ALIGNED) Les procédures relatives à la variante ALIGNED sont spécifiées dans les 10.9.3.1 à 10.9.3.8.4. (Les procédures relatives à la variante UNALIGNED sont spécifiées dans le 10.9.4.)

10.9.3.1 A la suite de l'analyse de la définition du type (spécifiée dans des paragraphes ultérieurs), le déterminant de longueur (qui est un nombre entier "n") aura été précisé sous une des formes suivantes:

a) une longueur normalement petite, avec une limite inférieure "lb" égale à 1;

b) un nombre entier contraint, avec une limite inférieure "lb" supérieure ou égale à zéro et une limite supérieure "ub" inférieure à 64K;

c) un nombre entier semi-contraint avec une limite inférieure "lb" supérieure ou égale à zéro, ou un nombre entier contraint avec une limite inférieure "lb" supérieure ou égale à zéro et une limite supérieure "ub" supérieure ou égale à 64K.

10.9.3.2 Les paragraphes invoquant les procédures du présent paragraphe auront déterminé au préalable une valeur pour la limite inférieure de longueur "lb" (qui est nulle si la longueur n'est pas contrainte) et pour la limite supérieure de longueur "ub". La valeur "ub" n'est pas fixée si aucune limite supérieure ne peut être déterminée sur la base des contraintes visibles par les règles PER.

10.9.3.3 Lorsque le déterminant de longueur est un nombre entier contraint dont la limite supérieure "ub" est inférieure à 64K, on doit annexer à la liste des champs le codage du nombre entier contraint représentant le déterminant de longueur, comme spécifié au 10.5. Si le nombre "n" est différent de zéro, il doit être suivi du champ associé ou de la liste des champs associés, ce qui met fin aux procédures du présent paragraphe. Si le nombre "n" est égal à zéro, on ne doit plus rien annexer à la liste des champs, ce qui met fin aux procédures du présent paragraphe.

NOTES

1 Par exemple:

A ::= Foo (SIZE (3..6)) -- *La longueur est codée dans un champ binaire de 2 bits*

B ::= Foo (SIZE (40000..40254)) -- *La longueur est codée dans un champ binaire de 8 bits*

C ::= Foo (SIZE (0..32000)) -- *La longueur est codée dans un champ binaire de 2 octets calé à l'octet*

D ::= Foo (SIZE (64000)) -- *La longueur n'est pas codée*

2 Le fait de ne plus rien annexer si "n" est égal à zéro a pour conséquence que le bourrage jusqu'à une limite d'octet ne se produit pas lorsque ces procédures sont invoquées pour annexer un champ binaire aligné en octets de longueur nulle, à moins que cela ne soit prescrit par le 10.5.

10.9.3.4 Lorsque le déterminant de longueur est un nombre normalement petit, avec "n" inférieur ou égal à 64, on doit annexer à la liste des champs un champ binaire ne comportant qu'un seul élément binaire forcé à 0 et on doit coder la valeur "n – 1" sous la forme d'un entier binaire non négatif dans un champ binaire de 6 éléments. Ce champ doit être suivi du champ associé, ce qui met fin aux procédures du présent paragraphe. Si le nombre "n" est supérieur à 64, on doit annexer à la liste des champs un champ binaire unitaire forcé à 1, suivi du codage de "n" sous la forme d'un déterminant de longueur semi-contraint, lui-même suivi du champ associé, conformément aux procédures des 10.9.3.5 à 10.9.3.8.4.

NOTE – Normalement, les champs courts sont uniquement utilisés pour indiquer la longueur de la phototrame qui précède les valeurs d'ajouts d'extensions de type ensemble ou séquence.

10.9.3.5 Dans les autres cas (nombre semi-contraint ou grande limite supérieure "ub"), le nombre "n" est codé et ajouté à la liste des champs, suivi des champs associés comme spécifié ci-dessous.

NOTE – La limite inférieure, "lb", n'a pas d'incidence sur les codages de longueur spécifiés aux 10.9.3.6 à 10.9.3.8.4.

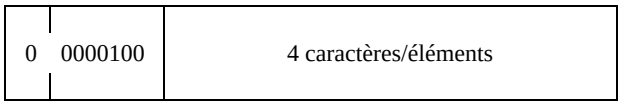
10.9.3.6 Si "n" est inférieur ou égal à 127, il doit être codé sous la forme d'un entier binaire non négatif (au moyen des procédures du 10.3) sur les bits 7 (le plus significatif) à 1 (le moins significatif) d'un seul octet dont le bit 8 sera forcé à zéro. Cet octet doit être ajouté à la liste des champs en tant que champ binaire calé à l'octet et être suivi du champ associé ou de la liste des champs associés, ce qui met fin aux procédures du présent paragraphe.

NOTE – Par exemple, si dans les productions suivantes la valeur de "A" a une longueur de quatre caractères et celle de "B" une longueur de quatre éléments,

A ::= IA5String

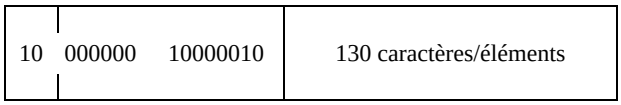
B ::= Foo (SIZE (4..123456))

les deux valeurs sont codées avec un octet de longueur occupant un octet, le premier bit étant mis à zéro pour indiquer que la longueur est inférieure ou égale à 127:



10.9.3.7 Si "n" est supérieur à 127 et inférieur à 16K, il doit être codé sous la forme d'un entier binaire non négatif (au moyen des procédures du 10.3) sur les bits 6 du premier octet (le plus significatif) à 1 du deuxième octet (le moins significatif) d'un champ binaire calé à l'octet composé de deux octets, le bit 8 du premier octet étant forcé à 1 et le bit 7 du premier octet étant forcé à 0. Ce champ doit être ajouté à la liste des champs et être suivi du champ associé ou de la liste des champs associés, ce qui met fin aux procédures du présent paragraphe.

NOTE – Si dans l'exemple du 10.9.3.6, "A" a une valeur de longueur 130 caractères et "B" une longueur de 130 éléments, les deux valeurs sont codées avec une composante de longueur occupant 2 octets, les deux premiers bits étant mis à 10 pour indiquer que la longueur est supérieure à 127 mais inférieure à 16K.



10.9.3.8 Si "n" est supérieur ou égal à 16K, on doit annexer à la liste des champs un seul octet d'un champ binaire calé à l'octet, dont le bit 8 est mis à 1 et le bit 7 mis à 1, les bits 6 à 1 codant pour la valeur 1, 2, 3 ou 4 sous la forme d'un entier binaire non négatif (au moyen des procédures du 10.3). Cet octet unique doit être suivi d'une partie du champ associé ou de la liste de champs associés, comme spécifié ci-dessous.

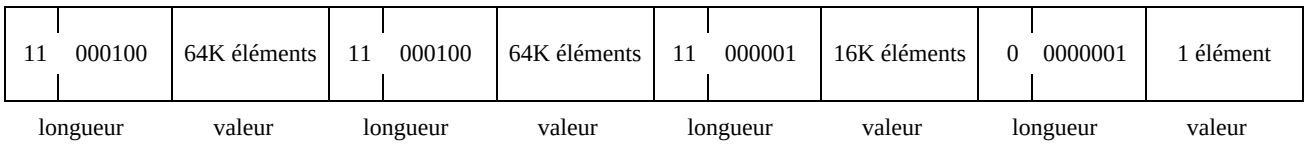
NOTE – La valeur des bits 6 à 1 est limitée aux nombres 1 à 4 (au lieu des limites théoriques 0 à 63), en sorte de limiter le nombre des éléments dont la réalisation doit avoir connaissance à un nombre plus maniable (64K au lieu de 1024K).

10.9.3.8.1 La valeur des bits 6 à 1 (soit 1 à 4) doit être multipliée par 16K pour obtenir un compte (appelé par exemple "m"). Le choix de l'entier représenté par les bits 6 à 1 doit être la valeur maximale admissible, telle que le champ associé ou la liste des champs associés contienne un nombre supérieur ou égal à "m" d'octets, de bits, de composantes ou de caractères, selon le cas.

NOTES

1 La forme non fragmentée permet des longueurs jusqu'à 16K. La fragmentation permet donc des longueurs pouvant atteindre 64K, avec une granularité de 16K.

2 Si dans l'exemple du 10.9.3.6 "B" a une valeur de longueur 144K + 1 (c'est-à-dire 64K + 64K + 16K + 1) éléments de longueur, cette valeur est fragmentée, les deux premiers bits des trois premiers fragments étant mis à 11 pour indiquer qu'un bloc sur quatre comporte 16K éléments, une autre composante de longueur suivant le dernier bloc de chaque fragment:



10.9.3.8.2 La partie du contenu spécifiée par "m" doit ensuite être ajoutée à la liste des champs, sous la forme d'un des champs suivants:

a) un champ binaire calé à l'octet comportant un seul octet codant pour les "m" premiers octets du champ associé, représentant des unités qui sont des octets;

b) un champ binaire calé à l'octet comportant un seul octet codant pour les "m" premiers bits du champ associé, représentant des unités qui sont des bits;

c) la liste des champs codant pour les "m" premières composantes de la liste des champs associés, représentant des unités qui sont des composantes d'un type ensemble-de ou séquence-de;

d) un champ binaire calé à l'octet comportant un seul octet codant pour les "m" premiers caractères du champ associé, représentant des unités qui sont des caractères.

10.9.3.8.3 Les procédures du 10.9 doivent ensuite être réappliquées afin d'ajouter la partie restante du champ associé ou de la liste des champs associés, sur une longueur qui est un nombre entier semi-contraint égal à ("n" – "m"), avec une limite inférieure égale à zéro.

NOTE – Si le dernier fragment qui contient une partie de la valeur codée a une longueur qui est un multiple exact de 16K, ce fragment est suivi d'un fragment final composé d'une composante de longueur sur un octet mis à 0.

10.9.3.8.4 L'adjonction d'une partie seulement du (des) champ(s) associé(s) à la liste des champs, suivie d'une réapplication des présentes procédures, est appelée **procédures de fragmentation**.

10.9.4 (Variante UNALIGNED) Les procédures pour la variante UNALIGNED sont spécifiées dans les 10.9.4.1 et 10.9.4.2. (Les procédures pour la variante ALIGNED sont spécifiées dans le 10.9.3.)

10.9.4.1 Si le déterminant de longueur "n" à coder est un nombre entier contraint dont la "plage" ("ub" – "lb" + 1) est inférieure à 64K, ce nombre "n" doit être codé sous la forme d'un entier binaire non négatif (comme spécifié au 10.3) au moyen du nombre minimal de bits nécessaire pour coder cette "plage". Si "n" est différent de zéro, il doit être suivi d'un champ associé ou d'une liste de champs associés, ce qui met fin aux procédures du présent paragraphe. Si "n" est égal à zéro, il ne doit pas y avoir d'autre adjonction à la liste des champs, ce qui met fin aux procédures du présent paragraphe.

NOTE – Si la "plage" satisfait à l'inégalité 2^m < "plage" <= 2^{m+1}, le nombre de bits contenus dans le déterminant de longueur est égal à m + 1.

10.9.4.2 Si le déterminant de longueur "n" à coder est un nombre entier contraint supérieur ou égal à 64K ou est un nombre entier semi-contraint, le nombre "n" doit être codé comme indiqué dans les 10.9.3.4 à 10.9.3.8.4.

11 Codage du type booléen

11.1 Une valeur du type booléen doit être codée sous la forme d'un champ binaire composé d'un seul élément.

11.2 Cet élément binaire doit être mis à 1 pour indiquer TRUE et à 0 pour indiquer FALSE.

11.3 Ce champ binaire doit être ajouté à la liste des champs, sans déterminant de longueur.

12 Codage du type entier

NOTES

1 (Didactique – Variante ALIGNED) Les plages qui permettent de coder toutes les valeurs dans un octet ou moins entrent dans un champ binaire de taille minimale, sans mesure de longueur. Les plages qui permettent de coder toutes les valeurs sur deux octets entrent dans un champ binaire de deux octets calé à l'octet, sans déterminant de longueur. Sinon, la valeur est codée sur le nombre minimal d'octets (sous la forme d'un entier binaire positif ou d'un entier binaire en complément à deux, selon ce qui convient) et on ajoute un déterminant de longueur. Dans ce cas, si la valeur de l'entier peut être codée sur moins de 127 octets (avec un décalage pouvant être déterminé par rapport à une éventuelle limite inférieure), et s'il n'existe ni limite inférieure ni limite supérieure, alors il y aura un déterminant de longueur d'un octet, autrement on codera la longueur sur le nombre minimal de bits nécessaire. Les autres cas n'ont pas d'intérêt pratique mais sont spécifiés pour mémoire.

2 (Didactique – Variante UNALIGNED) Les entiers contraints sont codés sur le plus petit nombre de bits nécessaire pour représenter la plage, quelle que soit l'étendue de cette plage. Les entiers non contraints sont codés conformément à la Note 1.

12.1 Si un marqueur d'extension est présent dans la spécification de contrainte du type entier, un seul élément binaire doit être ajouté à la liste des champs, sous la forme d'un champ binaire d'un bit de longueur. Ce bit doit être mis à 1 si la valeur à coder n'est pas dans la plage de la racine d'extension; dans le cas contraire, il doit être mis à 0. Dans le premier cas, la valeur doit être ajoutée à la liste des champs sous la forme d'une valeur d'entier non contrainte, comme spécifié dans les 12.2.4 à 12.2.6, ce qui met fin aux procédures du présent paragraphe. Dans le deuxième cas, la valeur doit être codée comme si le marqueur d'extension n'était pas présent.

12.2 Si un marqueur d'extension n'est pas présent dans la spécification de contrainte du type entier, les règles suivantes s'appliquent:

12.2.1 Si des contraintes visibles par les règles PER restreignent la valeur du type entier à une seule valeur, il ne doit pas y avoir d'adjonction à la liste des champs, ce qui met fin aux procédures du présent paragraphe.

12.2.2 Si des contraintes visibles par les règles PER restreignent la valeur du type entier à un nombre entier contraint, celui-ci doit être converti en un champ conformément aux procédures du 10.5 (codage d'un nombre entier contraint) et les procédures des 12.2.5 et 12.2.6 doivent ensuite être appliquées.

12.2.3 Si des contraintes visibles par les règles PER restreignent la valeur du type entier à un nombre entier semi-contraint, celui-ci doit être converti en un champ conformément aux procédures du 10.7 (codage d'un nombre entier semi-contraint) et les procédures du 12.2.6 doivent ensuite être appliquées.

12.2.4 Si des contraintes visibles par les règles PER ne restreignent pas le type entier à une valeur contrainte ou semi-contrainte de nombre entier, celui-ci doit être converti en un champ conformément aux procédures du 10.8 (codage d'un nombre entier non contraint) et les procédures du 12.2.6 doivent ensuite être appliquées.

12.2.5 Si les procédures invoquées pour coder la valeur du type entier dans un champ n'ont pas abouti au cas d'une longueur non définie (voir les 10.5.7.4 et 10.8.2), ce champ doit être ajouté à la liste des champs, ce qui met fin aux procédures du présent paragraphe.

12.2.6 Autrement (cas de la longueur non définie) on doit appliquer les procédures du 10.9 pour annexer le champ à la liste des champs, en le faisant précéder de l'un des déterminants de longueur suivants:

a) un déterminant de longueur contrainte "len" (tel que déterminé au 10.5.7.4) si des contraintes visibles depuis les règles PER imposent aux types des limites supérieure et inférieure et, dans le cas où le type est extensible, si la valeur appartient à la plage de la racine d'extension. La limite inférieure "lb" utilisée dans le déterminant de longueur sera prise à 1, et la limite supérieure "ub" correspondra au nombre d'octets nécessaire pour représenter la longueur de l'intervalle d'entiers.

NOTE – Le codage de la valeur "foo INTEGER (256..1234567) ::= 256" ne devrait donc pas être 00xxxxxx00000000, où chaque 'x' représente un bit de bourrage à zéro qui peut figurer ou non, selon l'emplacement de la longueur dans l'octet (par exemple, le codage sera 00 xxxxxx 00000000, si la longueur commence à une limite d'octet et 00 00000000, si elle commence au deuxième bit avant la fin d'un octet).

b) un déterminant de longueur non contrainte égal à "len" (tel que déterminé aux 10.7 et 10.8) si des contraintes visibles d'après les règles PER n'imposent pas aux types, des limites supérieure et inférieure finies, ou si le type est extensible et que la valeur ne se trouve pas à l'intérieur de la plage de la racine d'extension.

13 Codage du type énuméré

NOTE – (Didactique) Un type énuméré sans marqueur d'extension est codé comme s'il s'agissait d'un entier contraint dont la contrainte de sous-typage ne contient pas de marqueur d'extension. Autrement dit, un type énuméré sera presque toujours, en pratique, codé sous la forme d'un champ binaire, sur le plus petit nombre de bits nécessaire pour exprimer chaque énumération. En présence d'un marqueur d'extension et si la valeur n'est pas dans la racine d'extension, ce type est codé sous la forme d'un nombre entier non négatif normalement petit.

13.1 Les énumérations de la racine d'énumération doivent être triées en ordre ascendant, selon leurs valeurs d'énumération et doivent ensuite être affectées d'un index d'énumération commençant par zéro pour la première énumération, un pour la deuxième et ainsi de suite, jusqu'à la dernière énumération de la liste triée. Un indice d'énumération doit être affecté aux additions d'extension (qui sont toujours définies par ordre croissant), cet indice commençant par la valeur zéro pour la première énumération, un pour la deuxième et ainsi de suite jusqu'à la dernière énumération des additions d'extension.

NOTE – La Rec. UIT-T X.680/Amd. 1 | ISO/CEI 8824-1/Amd. 1 prescrit que les adjonctions successives d'extensions doivent avoir des valeurs d'énumération croissantes.

13.2 Si le marqueur d'extension ne figure pas dans la définition du type énuméré, l'index d'énumération doit être codé, comme s'il s'agissait d'une valeur d'entier dont la contrainte de type ne contenait pas de marqueur d'extension, la limite inférieure étant 0 et la limite supérieure, le plus grand index d'énumération associé au type, ce qui met fin aux procédures du présent paragraphe.

13.3 Si le marqueur d'extension figure, un seul bit doit être ajouté à la liste des champs, dans un champ binaire d'un élément de longueur. Ce bit doit être mis à 1 si la valeur à coder n'est pas dans la racine d'extension et à 0 dans le cas contraire. Dans le premier cas, les adjonctions à une énumération doivent être triées conformément au 13.1 et la valeur doit être ajoutée à la liste des champs sous la forme d'un nombre entier non négatif dont la valeur, normalement petite, est l'index d'énumération de l'énumération additionnelle et dont la limite "lb" est mise à 0, ce qui met fin aux procédures du présent paragraphe. Dans le deuxième cas, la valeur doit être codée comme si le marqueur d'extension n'était pas présent, comme spécifié au 13.2.

NOTE – Il n'existe pas de contrainte, visible par les règles PER, applicable à un type énuméré.

14 Codage du type réel

NOTE – (Didactique) Un type réel utilise les octets de champ de contenu selon les règles DER ou CER, précédés d'un déterminant de longueur qui, en pratique, sera un octet unique.

14.1 Si la base de la valeur abstraite est 10, la base de la valeur codée doit être 10; si la base de la valeur abstraite est 2, la base de la valeur codée doit être 2.

14.2 Le codage de REAL, spécifié pour les règles de codage canonique et pour les règles de codage distinctif dans la Rec. UIT-T X.690 | ISO/CEI 8825-1 doit être appliqué pour obtenir un champ binaire calé à l'octet qui est le contenu du codage CER/DER. Le contenu de ce codage consiste en "n" (par exemple) octets, et est placé dans un champ binaire calé à l'octet de "n" octets. Les procédures du 10.9 doivent être appliquées pour accoler ce champ calé à l'octet de "n" octets à la liste des champs, précédé d'un déterminant de longueur non contrainte égal à "n".

15 Codage du type chaîne binaire

NOTE – (Didactique) Les chaînes binaires de longueur contrainte inférieure ou égale à deux octets n'exigent pas d'alignement en octets, contrairement aux chaînes binaires plus longues. Si la longueur est fixée par des contraintes et que la limite supérieure soit inférieure à 64K, il n'y a pas de codage explicite de longueur; si ce n'est pas le cas, un codage de longueur est inclus sous l'une des formes spécifiées ci-dessus pour les codages de longueur, y compris les procédures de fragmentation de longues chaînes binaires.

15.1 Les contraintes visibles par les règles PER ne peuvent s'exercer que sur la longueur des chaînes binaires.

15.2 S'il n'y a pas de contraintes visibles par les règles PER et que le 19.7 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 s'applique, la valeur doit être codée sans bits de fin à zéro (c'est-à-dire qu'une valeur sans bits à 1 est toujours codée comme une chaîne binaire vide).

15.3 Lorsqu'il existe une contrainte visible par les règles PER et que le 19.7 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 s'applique (c'est-à-dire que le type de chaîne binaire est défini au moyen d'une liste de bits nommés "NamedBitList"), la valeur doit être codée avec adjonction ou extraction de bits de fin à zéro selon ce qui est nécessaire pour faire en sorte que la taille de la valeur transmise soit la plus petite compatible avec le transport de cette valeur tout en satisfaisant à la contrainte effective de taille.

15.4 Soit "ub" le nombre maximal de bits dans la chaîne binaire et "lb" le nombre minimal de bits (déterminés selon les contraintes visibles par les règles PER concernant la longueur). S'il n'existe pas de nombre maximal fini, on déclare que la limite "ub" n'est pas fixée. S'il n'existe pas de contrainte sur le nombre minimal, la limite "lb" a la valeur zéro. Soit "n" bits la longueur de la valeur réelle de chaîne binaire à coder.

15.5 Si un marqueur d'extension est présent dans la spécification de contrainte de taille du type chaîne binaire, un seul bit doit être ajouté à la liste des champs, sous la forme d'un champ binaire de longueur unitaire. Ce bit doit être mis à 1 si la longueur de ce codage n'est pas à l'intérieur de la plage de la racine d'extension, sinon, il doit être mis à 0. Dans le premier cas, le 15.10 doit être invoqué pour annexer la longueur à la liste des champs, sous la forme d'un nombre entier semi-contraint, suivi de la valeur de chaîne binaire. Dans le deuxième cas, la longueur et la valeur doivent être codées comme si le marqueur d'extension n'était pas présent.

15.6 Si aucun marqueur d'extension ne figure dans la spécification de contrainte du type chaîne binaire, les règles des 15.7 à 15.10 s'appliquent.

15.7 Si la chaîne binaire est contrainte à avoir une longueur zéro ("ub" = 0), cette chaîne ne doit pas être codée (et aucun champ n'est ajouté à la liste de champs), ce qui met fin aux procédures du présent paragraphe.

15.8 Si toutes les valeurs de la chaîne binaire sont contraintes à être de la même longueur ("ub" = "lb") et que cette longueur soit inférieure ou égale à 16 éléments binaires, cette chaîne binaire doit être placée dans un champ binaire de longueur contrainte "ub"; ce champ doit être ajouté à la liste des champs sans déterminant de longueur, ce qui met fin aux procédures du présent paragraphe.

15.9 Si toutes les valeurs de la chaîne binaire sont soumises à la contrainte d'être de la même longueur ("ub" = "lb") et que cette longueur est supérieure à 16 bits mais inférieure à 64K bits, la chaîne binaire doit être placée dans un champ binaire calé à l'octet de longueur "ub" (qui n'est pas nécessairement un multiple de 8 bits); ce champ doit être annexé à la liste des champs sans déterminant de longueur, ce qui met fin aux procédures du présent paragraphe.

15.10 Si les 15.7 à 15.9 ne s'appliquent pas, la chaîne binaire doit être placée dans un champ binaire calé à l'octet de longueur "n" bits et les procédures du 10.9 doivent être invoquées pour annexer ce champ binaire calé à l'octet de "n" bits à la liste des champs, précédé d'un déterminant de longueur égal à "n" bits sous la forme d'un nombre entier contraint si la limite "ub" est fixée et est inférieure à 64K ou sous la forme d'un nombre entier semi-contraint si la limite "ub" n'est pas fixée. La limite "lb" est déterminée comme ci-dessus.

NOTE – La fragmentation s'applique à un nombre non contraint ou à une limite supérieure "ub" élevée, au-delà de 16K, 32K, 48K ou 64K bits.

16 Codage du type chaîne d'octets

NOTE – Les chaînes d'octets de longueur fixe inférieure ou égale à deux octets ne sont pas alignées en octets, contrairement aux chaînes d'octets plus longues. La longueur des chaînes d'octets non contraintes est explicitement codée (si nécessaire avec fragmentation).

16.1 Les contraintes visibles par les règles PER ne peuvent limiter que la longueur de la chaîne d'octets.

16.2 Soit "ub" le nombre maximal d'octets dans la chaîne d'octets et "lb" le nombre minimal d'octets (déterminés selon les contraintes visibles par les règles PER concernant la longueur). S'il n'existe pas de nombre maximal fini, on déclare que la limite "ub" n'est pas fixée. S'il n'existe pas de contrainte sur le nombre minimal, la limite "lb" a la valeur zéro. Soit "n" octets la longueur de la valeur réelle de chaîne d'octets à coder.

16.3 Si une contrainte relative à la longueur, visible par les règles PER, est présente et qu'elle contient un marqueur d'extension, un seul bit doit être ajouté à la liste des champs, sous la forme d'un champ binaire de longueur unitaire. Ce bit doit être mis à 1 si la longueur de ce codage ne s'inscrit pas dans la plage de la racine d'extension; sinon, il doit être mis à 0. Dans le premier cas, le 16.8 doit être invoqué pour ajouter la longueur à la liste des champs, sous la forme d'un nombre entier semi-contraint, suivi de la valeur de chaîne d'octets. Dans le deuxième cas, la longueur et la valeur doivent être codées comme si le marqueur d'extension ne figurait pas.

16.4 Si aucun marqueur d'extension ne figure dans la spécification de contrainte du type chaîne d'octets, les règles des 16.5 à 16.8 s'appliquent.

16.5 Si la chaîne d'octets est soumise à la contrainte d'avoir une longueur zéro ("ub" = 0), cette chaîne ne doit pas être codée (et aucun champ n'est annexé à la liste de champs), ce qui met fin aux procédures du présent paragraphe.

16.6 Si toutes les valeurs de la chaîne d'octets sont soumises à la contrainte d'être de la même longueur ("ub" = "lb") et que cette longueur soit inférieure ou égale à deux octets, cette chaîne d'octets doit être placée dans un champ binaire de longueur contrainte "ub" multipliée par huit; ce champ doit être ajouté à la liste des champs sans déterminant de longueur, ce qui met fin aux procédures du présent paragraphe.

16.7 Si toutes les valeurs de la chaîne d'octets sont soumises à la contrainte d'être de la même longueur ("ub" = "lb") et que cette longueur soit supérieure à deux octets mais inférieure à 64K, la chaîne d'octets doit être placée dans un champ binaire calé à l'octet de longueur contrainte "ub"; ce champ doit être ajouté à la liste des champs sans déterminant de longueur, ce qui met fin aux procédures du présent paragraphe.

16.8 Si les 16.5 à 16.7 ne s'appliquent pas, la chaîne d'octets doit être placée dans un champ binaire calé à l'octet de longueur "n" octets et les procédures du 10.9 doivent être invoquées pour annexer à la liste des champs ce champ binaire de "n" octets alignés, précédé d'un déterminant de longueur égal à "n" sous la forme d'un entier contraint si la limite "ub" est fixée, ou sous la forme d'un entier semi-contraint si elle ne l'est pas. La limite "lb" est déterminée comme ci-dessus.

NOTE – Les procédures de fragmentation peuvent s'appliquer au-delà de 16K, 32K, 48K ou 64K octets.

17 Codage du type néant

NOTE – (Didactique) Le type néant est essentiellement une marque de réservation, qui ne prend un sens que dans le cas d'un choix ou d'une composante facultative d'un ensemble ou d'une séquence. L'identification du type néant dans un choix, ou sa présence en tant qu'élément facultatif, est déterminée dans les présentes règles de codage sans qu'il soit nécessaire d'avoir des octets pour représenter le type néant. Les valeurs nulles n'apporteront donc aucun octet à un codage.

Il ne doit pas y avoir d'addition à la liste des champs pour une valeur nulle.

18 Codage du type séquence

NOTE – (Didactique) Un type séquence commence par un préambule qui est une table de mappage binaire. Si le type séquence ne possède pas de marqueur d'extension, la table de mappage binaire précise simplement la présence ou l'absence de composantes par défaut ou facultatives dans ce type, codées sous la forme d'un champ binaire de longueur fixe. Si le type séquence possède un marqueur d'extension, la table de mappage binaire est précédée d'un seul bit qui indique si le codage contient effectivement des adjonctions d'extension. Le préambule est codé sans déterminant de longueur, à condition que celle-ci soit inférieure à 64K bits; sinon, un déterminant de longueur est codé afin de la segmenter. Le préambule est suivi des champs qui représentent chacune des composantes successives. S'il y a des adjonctions d'extension, on insérera immédiatement avant l'indicateur de la première adjonction un nombre entier semi-contraint codant le nombre d'adjonctions d'extension au type à coder; ce nombre est suivi d'une table de mappage binaire de longueur égale à ce nombre, indiquant la présence ou l'absence de valeurs pour chaque adjonction d'extension. Cette table est suivie des codages d'adjonction d'extension comme si chacun était la valeur d'un champ de type ouvert.

18.1 Si le type séquence possède un marqueur d'extension, un seul bit doit d'abord être ajouté à la liste des champs, sous forme d'un champ binaire d'un élément de longueur. Ce bit doit être mis à 1 si ce codage contient des valeurs d'adjonction d'extension et à 0 dans le cas contraire. (Ce bit sera appelé "bit d'extension" dans la suite du texte.) S'il n'y a pas de marqueur d'extension, aucun bit d'extension n'est ajouté.

18.2 Si le type séquence possède "n" composantes dans la racine d'extension et que ces composantes soient marquées des valeurs OPTIONAL ou DEFAULT, un seul champ binaire de "n" bits doit être établi pour adjonction à la liste des champs. Les bits de ce champ binaire doivent, pris dans l'ordre, coder la présence ou l'absence d'un codage pour chaque composante facultative ou par défaut du type séquence. Une valeur binaire 1 doit coder pour la présence du codage de la composante et une valeur binaire 0 doit coder pour l'absence du codage de la composante. Le bit initial du préambule doit coder pour la présence ou l'absence de la première composante facultative ou par défaut et le bit final doit coder pour la présence ou l'absence de la dernière composante facultative ou par défaut.

18.3 Si "n" est inférieur à 64K, le champ binaire doit être ajouté à la liste des champs. Si "n" est supérieur ou égal à 64K, les procédures du 10.9 doivent être invoquées pour annexer ce champ binaire de "n" bits à la liste des champs, précédé d'un déterminant de longueur égal à "n" bits sous forme de nombre entier contraint dont les limites "ub" et "lb" sont toutes deux rendues égales à "n".

NOTE – Dans ce cas, les limites "ub" et "lb" seront ignorées par les procédures relatives à la longueur. Ces procédures sont invoquées ici afin de permettre la fragmentation d'un long préambule. Il est considéré comme probable que cette situation ne se produira que rarement.

18.4 Le préambule doit être suivi des listes de champs de chacune des composantes de la valeur de séquence qui sont présentes, prises tour à tour.

18.5 Pour l'algorithme CANONICAL-PER, les codages des composantes marquées DEFAULT ne doivent jamais figurer si la valeur à coder est la valeur par défaut. Pour l'algorithme BASIC-PER, les codages des composantes marquées DEFAULT ne doivent jamais figurer si la valeur à coder est la valeur par défaut d'un type simple (voir 3.8.25); sinon elle ne devra être codée que si et seulement si elle figure explicitement dans la valeur de la séquence abstraite.

18.6 Ces règles mettent fin au codage si le bit d'extension est absent ou égal à zéro. S'il est présent et mis à 1, les procédures suivantes s'appliquent.

18.7 Soit "n" le nombre d'adjonctions d'extension au type à coder. Un champ binaire de "n" bits sera établi pour adjonction à la liste des champs. Les éléments de ce champ binaire doivent, pris dans l'ordre, coder la présence ou l'absence d'un codage pour chaque adjonction d'extension au type à coder. Une valeur binaire 1 doit coder pour la présence du codage de l'adjonction d'extension et une valeur binaire 0 doit coder pour l'absence du codage de l'adjonction d'extension. Le bit initial du champ binaire doit coder pour la présence ou l'absence de la première adjonction d'extension et le bit final doit coder pour la présence ou l'absence de la dernière adjonction d'extension.

18.8 Les procédures du 10.9 doivent être invoquées pour annexer ce champ binaire de "n" bits à la liste des champs, précédé d'un déterminant de longueur égal à "n" sous forme d'un nombre de longueur normalement petite.

NOTE – Le nombre "n" ne peut pas être nul car cette procédure n'est invoquée que s'il y a au moins une adjonction d'extension à coder.

18.9 Ces champs doivent être suivis de listes de champs contenant les codages de chaque adjonction d'extension présente, chacune étant prise à son tour et codée comme si elle était la valeur d'un champ de type ouvert, comme spécifié au 10.2.1.

19 Codage du type séquence-de

19.1 Des contraintes visibles par les règles PER peuvent contraindre le nombre de composantes du type séquence-de.

19.2 Soit "ub" le nombre maximal de composantes et "lb" le nombre minimal de composantes dans le type séquence-de (déterminés par les contraintes visibles par les règles PER). S'il n'y a pas de maximum fini ou si la limite "ub" est supérieure ou égale à 64K, on considère que la limite "ub" n'est pas fixée. S'il n'existe pas de contrainte sur le nombre minimal, la limite "lb" a la valeur zéro. Soit "n" le nombre de composantes dans la valeur réelle du type séquence-de à coder.

19.3 Le codage de chaque composante du type séquence-de produira un certain nombre de champs à annexer à la liste des champs pour le type séquence-de.

19.4 S'il y a une contrainte visible par les règles PER et contenant un marqueur d'extension, un seul bit doit être ajouté à la liste des champs, dans un champ binaire d'un élément de longueur. Ce bit doit être mis à 1 si le nombre de composantes de ce codage ne s'inscrit pas dans la plage de la racine d'extension et être mis à 0 dans le cas contraire.

Dans le premier cas, on doit invoquer le 19.7 pour ajouter la longueur à la liste des champs sous forme de nombre entier semi-contraint, suivi des valeurs de composante. Dans le deuxième cas, la longueur et la valeur doivent être codées comme si le marqueur d'extension n'était pas présent.

19.5 Si le nombre de composantes est fixe ("ub" = "lb") et que la limite "ub" soit inférieure à 64K, il ne doit pas y avoir de déterminant de longueur pour le type séquence-de et les champs de chaque composante doivent être ajoutés tour à tour à la liste des champs du type séquence-de.

19.6 Sinon, les procédures du 10.9 doivent être invoquées afin d'ajouter, à la liste des champs, la liste de tous les champs produits par les "n" composantes, précédée d'un déterminant de longueur égal à "n" composantes, sous forme de nombre entier contraint si la limite "ub" est fixée ou sous forme d'un nombre entier semi-contraint si la limite "ub" n'est pas fixée. La limite "lb" est déterminée comme ci-dessus.

NOTES

1 Les procédures de fragmentation peuvent s'appliquer après des composantes de 16K, 32K, 48K ou 64K.

2 Les points de séparation de la fragmentation sont situés entre les champs. Le nombre de bits précédant un point de séparation n'est pas forcément un multiple de huit.

20 Codage du type ensemble

Le type ensemble doit avoir les éléments de sa "RootComponentTypeList" triés dans l'ordre canonique spécifié au 6.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 et en outre (afin de déterminer l'ordre dans lequel les composantes seront codées si une ou plusieurs composantes sont du type choix non étiqueté) chaque type choix non étiqueté doit être ordonné comme s'il avait une étiquette de valeur égale à la plus petite étiquette de la "RootAlternativeTypeList" de ce type choix ou d'un quelconque des types choix non étiquetés pouvant y être imbriqués. Les éléments de l'ensemble qui figure dans la "AdditionalComponentTypeList" doivent être triés selon l'ordre canonique prescrit au 6.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1 indépendamment de la manière dont les types nommés de la liste ont été triés. La valeur du type ensemble sera alors codée comme si elle avait été déclarée avec le type séquence.

EXEMPLE – On suppose ici un environnement d'étiquetage du type IMPLICIT TAGS

```
A ::= SET
{
    a    [3] INTEGER,
    b    [1] CHOICE
    {
        c    [2] INTEGER,
        d    [4] INTEGER
    },
    e    CHOICE
    {
        f    CHOICE
        {
            g    [5] INTEGER,
            h    [6] INTEGER
        },
        i    CHOICE
        {
            j    [0] INTEGER
        }
    }
}
```

Dans cet exemple, l'ordre de codage des composantes de l'ensemble sera toujours e, b, a, car l'étiquette [0] aura la position correspondant à la plus petite valeur, suivie de [1], puis [3].

NOTE – La Rec. UIT-T X.680/Amd. 1 | ISO/CEI 8824-1/Amd. 1 prescrit que les adjonctions d'extension successives doivent avoir une valeur d'étiquette supérieure à celle de la dernière ajoutée à la "AdditionalComponentTypeList".

21 Codage du type ensemble-de

21.1 Pour l'algorithme CANONICAL-PER, les codages des valeurs des composantes du type ensemble-de doivent apparaître dans l'ordre ascendant, les codages de composante étant comparés sous forme de chaînes binaires, complétées avec jusqu'à sept bits de bourrage par 0 pour atteindre une limite d'octet.

NOTE – Aucun bit de bourrage, ajouté afin d'obtenir l'alignement en octets pour le tri, ne doit apparaître dans le codage final.

21.2 Pour l'algorithme BASIC-PER, le type ensemble-de doit être codé comme si l'on avait déclaré un type séquence-de.

22 Codage du type choix

NOTE – (Didactique) Un type choix est codé sous la forme d'un index spécifiant la variante choisie. Celle-ci est codée comme pour un entier contraint (à moins que le marqueur d'extension ne soit présent dans le type choix, auquel cas il s'agira d'un nombre entier non négatif normalement petit) et elle occupe donc, normalement, un champ binaire de longueur fixe, ayant le nombre minimal de bits nécessaire pour coder cet index (bien que ce champ puisse, en principe, avoir n'importe quelle longueur). Ce champ sera suivi du codage de la variante choisie. Les variantes qui sont des adjonctions d'extension seront codées comme si elles étaient des valeurs d'un champ de type ouvert. Lorsque le type choix ne possède qu'une seule variante d'option, il n'y a pas de codage pour l'index.

22.1 Le codage des types choix n'est pas influencé par les contraintes visibles par les règles PER.

22.2 A chaque composante d'un choix est associé un index dont la valeur est zéro pour la première variante contenue dans la racine du choix (les variantes étant considérées dans l'ordre canonique spécifié dans le 6.4 de la Rec. UIT-T X.680 | ISO/CEI 8824-1), un pour la deuxième variante et ainsi de suite jusqu'à la dernière composante contenue dans la racine d'extension du type choix. Une valeur d'index sera, de même, attribuée à chaque adjonction d'extension, en commençant par 0, exactement comme pour les composantes de racine d'extension. La valeur du plus grand index contenu dans la racine est "n" .

NOTE – La Rec. UIT-T X.680/Amd. 1 | ISO/CEI 8824-1/Amd. 1 prescrit que les adjonctions d'extension au type choix doivent chacune avoir une valeur d'étiquette supérieure à celle de la dernière ajoutée à la "AdditionalAlternativeTypeList".

22.3 En vue de l'ordonnancement canonique des variantes de choix contenant un type choix non étiqueté, chacun de ces types doit être ordonné comme s'il avait une étiquette de valeur égale à celle de la plus petite étiquette de la racine d'extension de ce type choix ou d'un quelconque des types choix non étiquetés pouvant y être imbriqués.

22.4 Si le choix ne comporte qu'une seule variante d'option dans la racine d'extension, il ne doit pas y avoir de codage de l'index si cette variante est choisie.

22.5 Si le type choix possède un marqueur d'extension, un seul bit doit d'abord être ajouté à la liste des champs, sous forme d'un champ binaire d'un seul élément de longueur. Ce bit doit être mis à 1 si le codage comporte une valeur d'adjonction d'extension, à 0 si ce n'est pas le cas. (Ce bit sera appelé ci-après "bit d'extension".) S'il n'y a pas de marqueur d'extension, aucun bit d'extension ne doit être ajouté.

22.6 Si le bit d'extension est absent, l'index de choix de la variante choisie doit être codé sous la forme d'un champ conformément aux procédures de l'article 12, comme s'il s'agissait d'une valeur de type entier (sans marqueur d'extension dans sa contrainte de sous-typage), cette valeur étant restreinte à la plage 0 à "n". Ce champ doit être ajouté à la liste des champs et être suivi des champs de la variante choisie, ce qui met fin aux procédures du présent paragraphe.

22.7 Si le bit d'extension est présent et que la variante choisie s'inscrive dans la racine d'extension, l'index de choix de la variante choisie doit être codé comme si le marqueur d'extension était absent, conformément à la procédure du 22.6, ce qui met fin aux procédures du présent paragraphe.

22.8 Si le bit d'extension est présent et que la variante choisie ne s'inscrive pas dans la racine d'extension, l'index de choix de la variante choisie doit être codé sous la forme d'un nombre entier non négatif normalement petit, avec la limite "lb" mise à 0; puis ce champ doit être ajouté à la liste des champs et être suivi d'une liste de champs contenant le codage de la variante choisie, codée comme si elle était la valeur d'un champ de type ouvert, comme spécifié au 10.2, ce qui met fin aux procédures du présent paragraphe.

23 Codage du type identificateur d'objet

NOTE – (Didactique) Le codage d'un type identificateur d'objet fait appel aux octets de champ de contenu prescrits par les règles BER, précédés d'un déterminant de longueur qui, en pratique, sera un octet unique.

Le codage spécifié par les règles BER doit être appliqué pour donner un champ binaire calé à l'octet qui est le champ de contenu du codage BER et qui se compose de (par exemple) "n" octets placés dans un champ binaire de "n" octets alignés. Les procédures du 10.9 doivent être invoquées pour annexer ce champ binaire calé à l'octet à la liste des champs, précédé d'un déterminant de longueur égal à "n", sous la forme d'un nombre entier semi-contraint mesurant le nombre d'octets.

24 Codage du type valeur PDV encapsulée

NOTE – (Didactique) Le type valeur PDV encapsulée est codé par un bit fanion unique comme en-tête, suivi d'un nombre entier non négatif normalement petit qui code pour un "indice" et qui est, en option, suivi d'une "identification" et, finalement, de la valeur codée. Le bit fanion détermine laquelle des deux règles secondaires sera appliquée dans le reste de la valeur PDV encapsulée. Ce fanion est mis à 1 pour indiquer l'utilisation de la règle secondaire "établissement de l'index" (EP-A); il est mis à 0 pour indiquer l'utilisation de la règle secondaire "utilisation de l'index" (EP-B). La règle secondaire "établissement de l'index" est utilisée lors de la première apparition d'une "identification" donnée dans la valeur de syntaxe abstraite dans laquelle le type valeur PDV encapsulée est défini. La règle secondaire "utilisation de l'index" est ensuite utilisée pour toutes les occurrences suivantes de cette "identification" dans les syntaxes abstraites dans lesquelles est défini le type valeur PDV encapsulée. Lorsque la règle secondaire "établissement de l'index" est utilisée, on code toute la valeur "identification". Il est prescrit qu'une valeur "index" apparaissant avec le fanion mis à 0 dans un codage de valeur PDV encapsulée apparaisse exactement une seule fois sous la forme d'une valeur "index" avec fanion mis à 1. Pour l'algorithme CANONICAL-PER, on n'utilisera que la variante "syntaxes" de la valeur "identification" (explicitement ou par l'intermédiaire de contraintes ASN.1 sur le type valeur PDV encapsulée). Il existe un autre moyen d'optimisation, qui est le suivant: si la syntaxe abstraite et la syntaxe de transfert obéissent toutes les deux à des contraintes monovaluantes visibles par les règles PER, ou si l'"identification" est soumise à la contrainte de la variante "fixe", le codage ne comprendra ni fanion à bit unique, ni "index" ni "identification".

24.1 Il y a trois façons de coder un type valeur PDV encapsulée:

a) la variante "syntaxes" du type valeur PDV encapsulée obéit à une contrainte monovaluante et est soumise à la contrainte de la variante "fixe", auquel cas seule la valeur de données doit être codée; cette option est dite "prédéfinie";

b) le premier bit de la valeur PDV encapsulée est mis à 1, auquel cas une valeur "index" doit être attribuée au préalable à la valeur "identification", qui doit être présente, suivie de l'élément "data-value" (valeur de données); cette option est appelée règle secondaire "établissement de l'index", ou "EP-A";

c) le premier bit de la valeur PDV encapsulée est mis à 0, auquel cas une valeur "index" et une "valeur de données" doivent être présentes mais en l'absence de valeur "identification"; cette option est appelée règle secondaire "utilisation de l'index", ou "EP-B".

24.2 L'option "prédéfinie" doit être utilisée soit lorsque "identification" est soumise à la contrainte de la variante "fixe", soit lorsque la variante "syntaxes" de l'"identification" est soumise à une contrainte monovaluante. Lorsque l'option "prédéfinie" est utilisée, le codage complet de l'élément "data-value" (valeur de données) doit être ajouté à un champ binaire calé à l'octet et le 10.9.3.5 doit être appliqué pour l'annexer à la liste des champs, précédé de la longueur, sous la forme d'un nombre entier semi-contraint, ce qui met fin aux procédures du présent paragraphe.

24.3 Lorsque l'option "prédéfinie" n'est pas sélectionnée, un bit unique doit être ajouté à la liste des champs, sous forme d'un champ binaire d'un élément de longueur, celui-ci étant mis à 1 si la règle secondaire "établissement de l'index" est utilisée, à 0 dans le cas contraire.

24.4 La règle secondaire "établissement de l'index" doit être utilisée la première fois que l'on spécifie une valeur abstraite pour "identification". Lorsque la règle secondaire "établissement de l'index" est utilisée, la procédure du 10.6 est invoquée pour ajouter une valeur d'index à la liste des champs, sous la forme d'un nombre entier non négatif et normalement petit. La première valeur d'index doit être 0 et doit augmenter d'une unité à chaque apparition d'une valeur "identification" distincte. L'article 22 sera ensuite invoqué pour ajouter cette "identification" à la liste des champs. Le codage complet de l'élément "data-value" (valeur de données) sera ensuite ajouté à un champ binaire calé à l'octet et le 10.9 sera invoqué pour l'annexer à la liste des champs après en avoir fixé la longueur sous la forme d'un nombre entier semi-contraint, ce qui met fin aux procédures du présent paragraphe.

24.5 La règle secondaire "utilisation de l'index" doit être appliquée à chaque apparition successive d'une valeur abstraite donnée pour l'élément "identification". Lorsque la règle secondaire "utilisation de l'index" est appliquée, on invoque les procédures du 10.6 pour ajouter une valeur d'index à la liste des champs, sous forme d'un nombre entier non négatif et normalement petit. Cette valeur d'index doit correspondre à celle qui a été préalablement ajoutée à la liste des champs pour cette valeur abstraite "identification". Le codage complet de l'élément "data-value" (valeur de données) sera ensuite ajouté à un champ binaire calé à l'octet et le 10.9 sera invoqué pour l'annexer à la liste des champs après en avoir fixé la longueur sous la forme d'un nombre entier semi-contraint, ce qui met fin aux procédures du présent paragraphe.

24.6 DIDACTIQUE: Ainsi, pour toute valeur "identification" donnée, il existe un codage (relativement peu efficace) par règle secondaire EP-A, contenant une unique valeur d'index ainsi que la valeur "identification" entière, suivies d'un nombre arbitraire de codages (efficaces) par règle secondaire EP-B, liés au codage EP-A par cette valeur d'index.

24.7 Le codage EP-A doit, selon les règles PER, fixer le fanion d'établissement ou d'utilisation d'index, suivi de l'index proprement dit et du type séquence suivant, pour lequel on a spécifié des étiquettes de type AUTOMATIC TAGS (voir 22.6 et 26.3 de la Rec. UIT-T X.680 | ISO/CEI 8824-1):

```
SEQUENCE {
    identification              CHOICE {
        syntaxes                    SEQUENCE {
            abstract                    OBJECT IDENTIFIER,
            transfer                    OBJECT IDENTIFIER },
        syntax                      OBJECT IDENTIFIER,
        presentation-context-id     INTEGER,
        context-negotiation         SEQUENCE {
            presentation-context-id     INTEGER,
            transfer-syntax             OBJECT IDENTIFIER },
        transfer-syntax             OBJECT IDENTIFIER,
        fixed                       NULL },
    data-value                  BIT STRING }
```

24.8 La valeur de "data-value" doit être le codage de la valeur de données abstraite, selon la syntaxe de transfert identifiée, et la valeur de chacun des autres champs doit être codée selon la même syntaxe de transfert que les valeurs apparaissant dans la valeur de la syntaxe abstraite.

NOTE – Les deux variantes de "data value" (valeur de données) de la syntaxe abstraite sont codées de façon identique – comme une chaîne binaire – dans la syntaxe de transfert.

25 Codage d'une valeur du type externe

25.1 Le codage d'une valeur du type externe doit être le codage PER du type séquence suivant, que l'on suppose avoir été défini dans un environnement étiqueté de type EXPLICIT TAGS, de valeur spécifiée dans les paragraphes ci-après:

```
[UNIVERSAL 8] IMPLICIT SEQUENCE {
    direct-reference            OBJECT IDENTIFIER OPTIONAL,
    indirect-reference          INTEGER OPTIONAL,
    data-value-descriptor       ObjectDescriptor OPTIONAL,
    encoding                    CHOICE {
        single-ASN1-type            [0] ABSTRACT-SYNTAX.&Type,
        octet-aligned               [1] IMPLICIT OCTET STRING,
        arbitrary                   [2] IMPLICIT BIT STRING } }
```

NOTE – Ce type séquence est identique à celui qui est spécifié dans la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990.

25.2 La valeur des composantes dépend de la valeur abstraite en cours de transmission, qui est une valeur du type spécifié au 30.5 de la Rec. UIT-T X.680 | ISO/CEI 8824-1.

25.3 L'élément "data-value-descriptor" (descripteur de valeur de données) ci-dessus ne doit être présent que s'il est également présent dans la valeur abstraite, avec la même valeur.

25.4 Les valeurs des éléments "direct-reference" et "indirect-reference" ci-dessus doivent être présentes ou absentes conformément au Tableau 1, qui met en correspondance les variantes de type externe de l'élément "identification" définies dans la Rec. UIT-T X.680 | ISO/CEI 8824-1, paragraphe 30.5, avec les composantes de type externe "direct-reference" et "indirect-reference", définies au 8.18.1.

Tableau 1 – Variantes de codage pour l'élément "identification"

identification	direct-reference	indirect-reference
syntaxes	*** CAS IMPOSSIBLE ***	*** CAS IMPOSSIBLE ***
syntax	syntax	ABSENT
presentation-context-id	ABSENT	presentation-context-id
context-negotiation	transfer-syntax	presentation-context-id
transfer-syntax	*** CAS IMPOSSIBLE ***	*** CAS IMPOSSIBLE ***
fixed	*** CAS IMPOSSIBLE ***	*** CAS IMPOSSIBLE ***

25.5 L'élément "data-value" (valeur de données) doit être codé conformément à la syntaxe de transfert indiquée par le codage et doit être placé dans une variante du choix offert (ci-dessus) par l'élément "encoding", comme spécifié ci-après.

25.6 Si l'élément "data-value" (valeur de données) est la valeur d'un type de données ASN.1 unique et que les règles de codage pour cette valeur de données soient les mêmes que pour le type de données "EXTERNAL", l'instance d'expédition doit être la variante "single-ASN1-type".

25.7 Si le codage de l'élément "data-value" (valeur de données), au moyen d'une règle agréée ou négociée, se présente sous la forme d'un nombre entier d'octets, l'instance d'expédition doit être la variante "octet-aligned".

NOTE – Une valeur de données qui est une série de types ASN.1 et pour laquelle la syntaxe de transfert spécifie une simple concaténation des chaînes d'octets produites par l'application des règles BER à chaque type ASN.1, s'inscrit dans cette catégorie et non dans celle du 25.6.

25.8 Si le codage de l'élément "data-value" (valeur de données), au moyen d'une règle agréée ou négociée, ne se présente pas sous la forme d'un nombre entier d'octets, l'alternative choisie dans la structure choix "Encoding" sera "arbitrary".

25.9 Si, dans la structure choix "Encoding", l'alternative choisie est "single-ASN1-type", ce type ASN.1 remplacera le type ouvert, avec une valeur égale à l'élément "data-value" (valeur de données) à coder.

NOTE – L'étendue des valeurs pouvant être prises par le type ouvert est déterminée par l'enregistrement de la valeur de type identificateur d'objet qui est associée à l'élément "direct-reference" et/ou de la valeur d'entier associée à l'élément "indirect-reference".

25.10 Si, dans la structure choix "Encoding", l'alternative choisie est "octet-aligned", l'élément "data-value" (valeur de données) sera codé conformément à la syntaxe de transfert agréée ou négociée et les octets résultants doivent former la valeur du type chaîne d'octets.

25.11 Si, dans la structure choix "Encoding", l'alternative choisie est "arbitrary", l'élément "data-value" (valeur de données) sera codé conformément à la syntaxe de transfert agréée ou négociée et le résultat doit former la valeur du type chaîne binaire.

26 Codage des types chaîne de caractères restreinte

NOTES

1 (Didactique – Variante ALIGNED) Les chaînes de caractères de longueur fixe inférieure ou égale à deux octets ne sont pas alignées à l'octet, contrairement aux autres chaînes d'octets. Les chaînes de caractères de longueur fixe se codent sans champ de longueur si elles sont plus courtes que 64K caractères. La longueur des chaînes d'octets non contraintes ou des chaînes de caractères contraintes de longueur supérieure à 64K – 1, est explicitement codée (si nécessaire avec fragmentation). Chaque caractère d'une chaîne de type NumericString, PrintableString, VisibleString (ISO646String), IA5String, BMPString et UniversalString est codé sur le nombre de bits qui est la plus petite puissance de deux pouvant représenter tous les caractères autorisés par la contrainte effective d'alphabet permis.

2 (Didactique – Variante UNALIGNED) Les chaînes de caractères ne sont pas alignées à l'octet. S'il n'y a qu'une seule valeur de longueur possible, il n'y a pas de codage de longueur si celle-ci est inférieure à 64K caractères. Pour les chaînes de caractères non contraintes ou pour les chaînes de caractères contraintes de longueur supérieure à 64K – 1, la longueur est explicitement codée (avec fragmentation si nécessaire). Chaque caractère d'une chaîne de type NumericString, PrintableString, VisibleString (ISO646String), IA5String, BMPString et UniversalString est codé sur le nombre de bits qui est la plus petite puissance de deux pouvant représenter tous les caractères autorisés par la contrainte effective d'alphabet permis.

3 (Didactique – Au sujet de la taille de chaque caractère codé) Le codage de chaque caractère dépend de la contrainte d'alphabet permis (voir 9.3.10), qui définit l'alphabet utilisé pour le type. Supposons que cet alphabet se compose d'un jeu de caractères ALPHA. Pour chaque type chaîne de caractères à multiplicateur connu (voir 3.8.16), il existe une valeur d'entier associée à chaque caractère, obtenue par référence à une certaine table de codes associée au type chaîne de caractères restreinte. L'ensemble des valeurs BETA (par exemple), correspondant au jeu de caractères ALPHA, servira à déterminer le codage à utiliser, comme suit. Le nombre de bits pour le codage de chaque caractère est déterminé uniquement par le nombre d'éléments, N, contenus dans l'ensemble BETA (ou ALPHA). Dans la variante UNALIGNED, c'est le plus petit nombre de bits qui peut coder la valeur N – 1 sous forme d'un entier binaire non négatif. Dans la variante ALIGNED, c'est le plus petit nombre de bits qui est une puissance de deux et qui peut coder la valeur N – 1. Soit B le nombre de bits choisi. Si chaque valeur de l'ensemble BETA peut être codée (sans transformation) sur B bits, cette valeur de BETA sert à représenter les caractères correspondants dans l'ensemble ALPHA. Sinon, les valeurs de l'ensemble BETA sont prises dans l'ordre ascendant et sont remplacées par les valeurs 0, 1, 2, etc., jusqu'à N – 1; ce sont ces valeurs qui seront utilisées pour représenter le caractère correspondant. En résumé, on utilise toujours le plus petit nombre de bits (pris jusqu'à la prochaine puissance de deux dans la variante ALIGNED). On préférera donc utiliser la valeur normalement associée à chaque caractère mais, si l'une quelconque de ces valeurs ne peut pas être codée sur le nombre minimal de bits, on fera appel à une condensation.

26.1 Les types chaîne de caractères restreinte suivants sont des types chaîne de caractères à multiplicateur connu: NumericString, PrintableString, VisibleString (ISO646String), IA5String, BMPString, UniversalString. Les contraintes effectives d'alphabet permis ne sont visibles par les règles PER que pour ces types.

26.2 La notation de contrainte effective de taille peut déterminer une limite supérieure "aub" pour la longueur de la chaîne de caractères abstraite. Sinon, la limite "aub" n'est pas fixée.

26.3 La notation de contrainte effective de taille peut déterminer une limite inférieure "alb" pour la longueur de la chaîne de caractères abstraite. Sinon, la limite "alb" est égale à zéro.

NOTE – Les contraintes visibles par les règles PER ne s'appliquent qu'aux chaînes de type chaîne de caractères à multiplicateur connu. Pour les autres types de chaînes de caractères restreintes, la limite "aub" sera non fixée et la limite "alb" sera égale à zéro.

26.4 Si le type est extensible aux codages PER (voir 9.3.15), on doit ajouter à la liste des champs un champ binaire composé d'un seul bit. Celui-ci doit être mis à zéro si la valeur est conforme à la racine d'extension et à un dans l'autre cas. Si la valeur est extérieure à la plage de la racine d'extension, le codage qui suit doit être effectué comme en l'absence de contraintes effectives de taille et d'alphabet permis.

NOTE – Seules les chaînes de type chaîne de caractères à multiplicateur connu sont extensibles aux codages PER. Les marqueurs d'extensibilité placés sur des chaînes de caractères d'autres types n'ont pas d'incidence sur le codage PER.

26.5 Ce paragraphe s'applique aux chaînes de caractères à multiplicateur connu. Le codage d'autres types de chaînes de caractères restreintes est spécifié au 26.6.

26.5.1 L'alphabet effectivement permis est défini comme étant celui qui est autorisé par la contrainte d'alphabet permis ou comme l'entier alphabet du type natif, s'il n'y a pas de contrainte d'alphabet permis.

26.5.2 Soit N le nombre de caractères contenus dans l'alphabet effectivement permis et B le plus petit entier tel que 2 à la puissance B soit plus grand ou égal à N. Soit B2 la plus petite puissance de deux qui est supérieure ou égale à B. Dans la variante ALIGNED, chaque caractère doit se coder sur B2 éléments binaires et, dans la variante UNALIGNED, sur B éléments binaires. Soit "b" le nombre de bits identifiés par cette règle.

26.5.3 Une valeur numérique "v" est associée à chaque caractère par référence à l'article 36 de la Rec. UIT-T X.680 | ISO/CEI 8824-1, comme suit. Pour une chaîne de type UniversalString, la valeur est celle qui a été utilisée pour déterminer l'ordre canonique selon le 36.4 (la valeur est alors dans la plage comprise entre 0 et $2^{32} – 1$). Pour les chaînes de type BMPString, la valeur est celle qui a été utilisée pour déterminer l'ordre canonique selon le 36.5 (la valeur est alors dans la plage comprise entre 0 et $2^{16} – 1$). Pour les chaînes de types NumericString, PrintableString, VisibleString et IA5String, la valeur est celle qui a été définie pour le codage selon l'ISO 646 du caractère correspondant. (La plage est comprise entre 0 et 127 pour les chaînes IA5String, entre 32 et 126 pour les chaînes VisibleString, entre 32 et 57 pour les chaînes NumericString et entre 32 et 122 pour les chaînes PrintableString. Dans le cas des chaînes IA5String et VisibleString, toutes les valeurs de la plage figurent, alors que dans le cas des chaînes NumericString et PrintableString, toutes les valeurs de la plage ne sont pas en usage.)

26.5.4 Soit "lb" la plus petite et "ub" la plus grande valeur de l'intervalle pour le jeu de caractères dans l'alphabet permis. Le codage d'un caractère sur "b" bits sera donc représenté par l'entier binaire non négatif de valeur "v", déterminée comme suit:

a) si la limite "ub" est inférieure ou égale à $2^b – 1$, la variable "v" est la valeur spécifiée ci-dessus; sinon

b) les caractères sont placés dans l'ordre canonique défini à l'article 36 de la Rec. UIT-T X.680 | ISO/CEI 8824-1. Le premier caractère dans l'ordre canonique est affecté de la valeur zéro et le caractère suivant est affecté d'une valeur qui est supérieure d'une unité à la valeur affectée au caractère précédent dans l'ordre canonique. Ces valeurs sont désignées par "v".

NOTE – L'alinéa a) ci-dessus ne peut jamais s'appliquer à un caractère de chaîne numérique contrainte ou non contrainte, qui se code toujours sur un nombre de bits inférieur ou égal à quatre, selon l'alinéa b).

26.5.5 On obtiendra le codage de la chaîne de caractères entière en codant chaque caractère (au moyen d'une valeur "v" appropriée) sous la forme d'un entier binaire non négatif, sur "b" bits qui doivent être concaténés afin de constituer un champ binaire qui est un multiple de "b" bits.

26.5.6 Si la limite "aub" est égale à la limite "alb" et est inférieure à 64K, le champ binaire doit être ajouté à la liste des champs sous la forme d'un champ calé à l'octet si "aub" fois "b" donne un résultat supérieur à 16; sinon, ce champ doit être ajouté sous la forme d'un champ binaire non calé à l'octet, ce qui met fin aux procédures du présent paragraphe.

26.5.7 Si la limite "aub" n'est pas égale à la limite "alb" ou est supérieure ou égale à 64K, la procédure du 10.9 doit être invoquée afin d'ajouter un déterminant de longueur où "n" mesure le nombre de caractères dans la chaîne, avec "alb" comme limite inférieure et "aub" comme limite supérieure du déterminant de longueur. Le champ binaire du 26.5.5 doit ensuite être ajouté sous la forme d'un champ calé à l'octet si "aub" fois "b" donne un résultat supérieur à 16; sinon, ce champ doit être ajouté sous la forme d'un champ binaire non calé à l'octet, ce qui met fin aux procédures du présent paragraphe.

26.6 Ce paragraphe s'applique à des chaînes de caractères qui ne sont pas du type chaîne de caractères à multiplicateur connu. Dans ce cas, les contraintes ne sont jamais visibles par les règles PER et le type ne peut jamais être extensible au codage PER.

26.6.1 Pour l'algorithme BASIC-PER, la référence placée au-dessous de l'arc "base encoding" désigne les règles BER. Pour l'algorithme CANONICAL-PER, cette référence désigne les règles CANONICAL-BER.

26.6.2 Le codage de base ("base encoding") doit être appliqué à la chaîne de caractères afin de donner un champ de "n" octets.

26.6.3 Le 10.9 doit être appliqué pour ajouter un déterminant de longueur non contrainte, la variable "n" mesurant le nombre d'octets. Le champ de "n" octets doit être ajouté sous la forme d'un champ binaire calé à l'octet, ce qui met fin aux procédures du présent paragraphe.

27 Codage du type chaîne de caractères non restreinte

NOTE – (Didactique) Le type chaîne de caractères non restreinte est généralement codé par un bit unique de fanion en tête, suivi d'un nombre entier non négatif et normalement petit codant un "index", suivi optionnellement d'une "identification" et, finalement, de l'élément "string-value" (valeur de chaîne). Ce bit unique de fanion détermine laquelle des deux règles secondaires sera appliquée dans le reste de la valeur de chaîne de caractères non restreinte. Ce fanion est mis à 1 pour indiquer l'utilisation de la règle secondaire "établissement de l'index" (EP-A); il est mis à 0 pour indiquer l'utilisation de la règle secondaire "utilisation de l'index". La règle secondaire "établissement de l'index" est utilisée dès la première apparition d'une "identification" dans la valeur de la syntaxe abstraite. A chaque apparition ultérieure de cette valeur "identification" dans la valeur de la syntaxe abstraite dans laquelle est défini le type chaîne de caractères non restreinte, la règle secondaire "utilisation de l'index" est utilisée. Lorsque la règle secondaire "établissement de l'index" est utilisée, on code toute la valeur "identification". Une valeur "index" apparaissant avec le fanion mis à 0 dans un codage de valeur de chaîne de caractères non restreinte doit apparaître exactement une seule fois sous la forme d'une valeur "index" avec fanion mis à 1. Pour l'algorithme CANONICAL-PER, on n'utilisera que la variante "syntaxes" de la valeur "identification" (explicitement ou par l'intermédiaire de contraintes ASN.1 sur le type chaîne de caractères non restreinte). Il existe un autre moyen d'optimisation, qui est le suivant: si la syntaxe abstraite et la syntaxe de transfert obéissent toutes deux à des contraintes visibles par les règles PER leur imposant d'être monovaluées, le codage ne comportera ni bit fanion, ni "index" ni "identification".

27.1 Il y a trois façons de coder un type chaîne de caractères non restreinte:

a) la variante "syntaxes" du type chaîne de caractères non restreinte obéit à une contrainte lui imposant d'être monovaluée, ou la valeur "identification" est restreinte à la valeur "fixe", auquel cas seule la valeur de données doit être codée; cette option est dite "prédéfinie";

b) le premier bit de la valeur de chaîne de caractères non restreinte est mis à 1, auquel cas une valeur "index" doit être attribuée au préalable à la valeur "identification", qui doit être présente, suivie de l'élément "string-value" (valeur de chaîne); cette option est appelée règle secondaire "établissement de l'index", ou "EP-A";

c) le premier bit de la valeur de chaîne de caractères non restreinte est mis à 0, auquel cas une valeur "index" et un élément "string-value" doivent être présents mais en l'absence de valeur "identification"; cette option est appelée règle secondaire "utilisation de l'index", ou "EP-B".

27.2 L'option "prédéfinie" doit être utilisée lorsque la valeur "identification" est soumise à la contrainte de la variante "fixe" ou lorsque la variante "syntaxes" de cette valeur "identification" est soumise à une contrainte monovaluante. Lorsque l'option "prédéfinie" est utilisée, le codage complet de l'élément "string-value" (valeur de chaîne) doit être ajouté à un champ binaire calé à l'octet et le 10.9.3.5 doit être invoqué pour l'annexer à la liste des champs, après en avoir fixé la longueur sous la forme d'un nombre entier semi-contraint, ce qui met fin aux procédures du présent paragraphe.

27.3 Lorsque l'option "prédéfinie" n'est pas sélectionnée, un bit unique doit être ajouté à la liste des champs, sous forme d'un champ binaire d'un élément de longueur, celui-ci étant mis à 1 si la règle secondaire "établissement de l'index" est utilisée, à 0 dans le cas contraire.

27.4 La règle secondaire "établissement de l'index" doit être utilisée la première fois que l'on spécifie une valeur abstraite pour "identification". Lorsque la règle secondaire "établissement de l'index" est utilisée, la procédure du 10.6 est invoquée pour ajouter une valeur d'index à la liste des champs, sous la forme d'un nombre entier non négatif généralement petit. La première valeur d'index doit être 0 et doit augmenter d'une unité à chaque apparition d'une valeur "identification" distincte. L'article 22 sera ensuite invoqué pour ajouter cette "identification" à la liste des champs. Le codage complet de l'élément "string-value" (valeur de chaîne) sera ensuite ajouté à un champ binaire calé à l'octet et le 10.9 sera invoqué pour l'annexer à la liste des champs après en avoir fixé la longueur sous la forme d'un nombre entier semi-contraint, ce qui met fin aux procédures du présent paragraphe.

27.5 La règle secondaire "utilisation de l'index" doit être appliquée à chaque apparition successive d'une valeur abstraite donnée pour l'élément "identification". Lorsque la règle secondaire "utilisation de l'index" est appliquée, on invoque les procédures du 10.6 pour ajouter une valeur d'index à la liste des champs, sous forme d'un nombre entier non

négatif et normalement petit. Cette valeur d'index doit correspondre à celle qui a été préalablement ajoutée à la liste des champs pour cette valeur abstraite "identification". Le codage complet de l'élément "string-value" (valeur de chaîne) sera ensuite ajouté à un champ binaire calé à l'octet et le 10.9 sera invoqué pour l'annexer à la liste des champs après en avoir fixé la longueur sous la forme d'un nombre entier semi-contraint, ce qui met fin aux procédures du présent paragraphe.

27.6 DIDACTIQUE: Ainsi, pour toute valeur "identification" donnée, il existe un codage (relativement peu efficace) par règle secondaire EP-A, contenant une unique valeur d'index ainsi que la valeur "identification" entière, suivies d'un nombre arbitraire de codages (efficaces) par règle secondaire EP-B, liés au codage EP-A par cette valeur d'index.

27.7 Le codage EP-A correspondra au codage PER du bit fanion d'établissement ou d'utilisation d'index, suivi de l'index proprement dit, puis du type séquence suivant pour lequel aura été spécifié un étiquetage automatique (voir 22.6 et 26.3 de la Rec. UIT-T X.680 | ISO/CEI 8824-1):

```
SEQUENCE {
    identification              CHOICE {
        syntaxes                    SEQUENCE {
            abstract                    OBJECT IDENTIFIER,
            transfer                    OBJECT IDENTIFIER },
        syntax                      OBJECT IDENTIFIER,
        presentation-context-id     INTEGER,
        context-negotiation         SEQUENCE {
            presentation-context-id     INTEGER,
            transfer-syntax             OBJECT IDENTIFIER },
        transfer-syntax             OBJECT IDENTIFIER,
        fixed                       NULL },
    string-value                OCTET STRING }
```

27.8 La valeur de "string-value" (valeur de chaîne) doit être le codage de la valeur de données abstraite, selon la syntaxe de transfert indiquée, la valeur de tous les autres champs devant être codée en utilisant la même syntaxe de transfert que celle des autres valeurs apparaissant dans la valeur de la syntaxe abstraite.

NOTE – Les deux variantes de "string-value" (valeur de chaîne) de la syntaxe abstraite sont codées de façon identique – comme une chaîne d'octets – dans la syntaxe de transfert.

28 Identificateurs d'objet pour syntaxes de transfert

28.1 Les règles de codage spécifiées dans la présente Recommandation | Norme internationale peuvent être citées en référence et être appliquées chaque fois qu'il est nécessaire de spécifier une représentation non ambiguë d'une chaîne binaire pour toutes les valeurs d'un type ASN.1 donné.

28.2 Les valeurs suivantes, de types identificateur d'objet et descripteur d'objet, sont assignées pour identifier et décrire les règles de codage spécifiées dans la présente Recommandation | Norme internationale:

Pour l'algorithme BASIC-PER, variante ALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) aligned (0)}

"Codage compact d'un unique type ASN.1 (règles de base et champs calés à l'octet)"

Pour l'algorithme BASIC-PER, variante UNALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) unaligned (1)}

"Codage compact d'un unique type ASN.1 (règles de base et champs non calés à l'octet)"

Pour l'algorithme CANONICAL-PER, variante ALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) aligned (0)}

"Codage compact d'un unique type ASN.1 (règles canoniques et champs calés à l'octet)"

Pour l'algorithme CANONICAL-PER, variante UNALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) unaligned (1)}

"Codage compact d'un unique type ASN.1 (règles canoniques et champs non calés à l'octet)"

28.3 Lorsqu'une norme d'application définit une syntaxe abstraite sous la forme d'un ensemble de valeurs abstraites, dont chacune est une valeur d'un certain type dénommé spécifiquement en notation ASN.1 et défini en utilisant la notation ASN.1, les valeurs d'identificateur d'objet spécifiées au 28.2 peuvent être utilisées avec ces noms en syntaxe abstraite pour identifier les syntaxes de transfert qui sont issues de l'application – aux types ASN.1 dénommés spécifiquement et utilisés pour définir la syntaxe abstraite – des règles de codage spécifiées dans la présente Recommandation | Norme internationale.

NOTE – En particulier, ces identifications des règles de codage peuvent apparaître dans le champ "transfer syntax name" (nom de la syntaxe de transfert) du protocole de présentation (Rec. UIT-T X.226 | ISO/CEI 8823-1).

28.4 Les noms spécifiés au 28.2 ne doivent pas être utilisés avec un nom de syntaxe abstraite pour identifier une syntaxe de transfert si les conditions du 28.3 ne sont pas remplies pour la définition de la syntaxe abstraite.

Annexe A

Exemples de codages

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe donne des exemples de l'utilisation des règles de codage compact (PER) spécifiées dans la présente Recommandation | Norme internationale, en donnant la représentation objet d'un enregistrement (hypothétique) "PersonnelRecord" défini en ASN.1.

A.1 Enregistrement qui n'utilise pas de contrainte appliquée aux sous-types

A.1.1 Description ASN.1 de la structure de l'enregistrement

La structure de l'enregistrement hypothétique "PersonnelRecord" ("dossier individuel") est décrite formellement en utilisant la syntaxe ASN.1 spécifiée dans la Rec. UIT-T X.680 | ISO/CEI 8824-1 pour la définition des types. Cette description est identique à celle de l'exemple défini dans l'Annexe A de la Rec. UIT-T X.690 | ISO/CEI 8825-1.

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET {
        name            Name,
        title           [0] VisibleString,
        number          EmployeeNumber,
        dateOfHire      [1] Date,
        nameOfSpouse    [2] Name,
        children        [3] IMPLICIT
            SEQUENCE OF ChildInformation DEFAULT {} }

ChildInformation ::= SET
        { name          Name,
          dateOfBirth   [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
      {givenName        VisibleString,
       initial          VisibleString,
       familyName       VisibleString}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString  -- YYYYMMDD
```

A.1.2 Description ASN.1 d'une valeur d'enregistrement

La valeur de l'enregistrement John Smith est décrite formellement ci-après en ASN.1.

```
{ name {givenName "John",initial "P",familyName "Smith"},
  title         "Director",
  number        51,
  dateOfHire            "19710917",
  nameOfSpouse          {givenName "Mary",initial "T",familyName "Smith"},
  children
      {{name {givenName "Ralph",initial "T",familyName "Smith"},
             dateOfBirth "19571111"},
      {name {givenName "Susan",initial "B",familyName "Jones"},
             dateOfBirth "19590717"}}}
```

A.1.3 Représentation selon l'algorithme ALIGNED PER de cette valeur d'enregistrement

La représentation de la valeur d'enregistrement donnée ci-dessus (après application de la variante ALIGNED des règles de codage compact définies dans la présente Recommandation | Norme internationale) est donnée ci-après. Le codage hexadécimal est suivi d'une description commentée de la représentation binaire de ce codage.

La longueur de ce codage est de 94 octets. A titre de comparaison, la valeur PersonnelRecord codée selon la variante UNALIGNED des règles PER est de 84 octets; selon les règles BER avec la forme de longueur définie, elle est d'au moins 136 octets et selon les règles BER avec la forme de longueur non définie, d'au moins 161 octets.

A.1.3.1 Représentation hexadécimale

```
80044A6F 686E0150 05536D69 74680133 08446972 6563746F 72083139 37313039
3137044D 61727901 5405536D 69746802 0552616C 70680154 05536D69 74680831
39353731 31313105 53757361 6E014205 4A6F6E65 73083139 35393037 3137
```

A.1.3.2 Représentation binaire

Pour faciliter la lecture de la représentation binaire des données, des lignes blanches séparent les champs logiquement associés (notamment les couples longueur/valeur); un saut de ligne sépare les champs; un espace sépare les caractères dans une chaîne de caractères et un "x" représente un bit de bourrage à zéro, qui est parfois utilisé pour caler les champs à l'octet.

1xxxxxxx	Bit de mappage = 1 indique que "children" figure
00000100	Longueur de name.givenName = 4
01001010 01101111 01101000 01101110	name.givenName = "John"
00000001	Longueur de name.initial = 1
01010000	name.initial = "P"
00000101	Longueur de name.familyName = 5
01010011 01101101 01101001 01110100 01101000	name.familyName = "Smith"
00000001	Longueur de (employee) number = 1
00110011	(employee) number = 51
00001000	Longueur de title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010	title = "Director"
00001000	Longueur de dateOfHire = 8
00110001 00111001 00110111 00110001 00110000 00111001 00110001 00111111	dateOfHire = "19590717"
00000100	Longueur de nameOfSpouse.givenName = 4
01001101 01100001 01110010 01111001	nameOfSpouse.givenName = "Mary"
00000001	Longueur de nameOfSpouse.initial = 1
01010100	nameOfSpouse.initial = "T"
00000101	Longueur de nameOfSpouse.familyName = 5
01010011 01101101 01101001 01110100 01101000	nameOfSpouse.familyName = "Smith"
00000010	Nombre d'enfants
00000101	Longueur de children[0].givenName = 5
01010010 01100001 01101100 01110000 01101000	children[0].givenName = "Ralph"
00000001	Longueur de children[0].initial = 1
01010100	children[0].initial = "T"
00000101	Longueur de children[0].familyName = 5
01010011 01101101 01101001 01110100 01101000	children[0].familyName = "Smith"
00001000	Longueur de children[0].dateOfBirth = 8
00110001 00111001 00110101 00110111 00110001 00110001 00110001 00110001	children[0].dateOfBirth = "19571111"
00000101	Longueur de children[1].givenName = 5
01010011 01110101 01110011 01100001 01101110	children[1].givenName = "Susan"
00000001	Longueur de children[1].initial = 1
01000010	children[1].initial = "B"
00000101	Longueur de children[1].familyName = 5
01001010 01101111 01101110 01100101 01110011	children[1].familyName = "Jones"
00001000	Longueur de children[1].dateOfBirth = 8
00110001 00111001 00110101 00111001 00110000 00110111 00110001 00110111	children[1].dateOfBirth = "19590717"

A.1.4 Représentation UNALIGNED PER de cette valeur d'enregistrement

La représentation de la valeur d'enregistrement donnée ci-dessus (après application de la variante UNALIGNED des règles de codage compact définies dans la présente Recommandation | Norme internationale) est représentée ci-dessous. Le codage hexadécimal est suivi d'une description commentée de la forme de codage binaire. A noter qu'aucun bit de bourrage ne figure dans la variante UNALIGNED et que les caractères sont codés avec le plus petit nombre de bits possible.

La longueur de ce codage est de 84 octets. A titre de comparaison, la valeur de PersonnelRecord codée selon la variante ALIGNED des règles PER est de 94 octets, dans la forme de longueur définie des règles BER elle est d'au moins 136 octets et dans la forme de longueur non définie des règles BER elle est d'au moins 161 octets.

A.1.4.1 Représentation hexadécimale

```
824ADFA3  700D005A  7B74F4D0  02661113  4F2CB8FA  6FE410C5  CB762C1C  B16E0937
0F2F2035  0169EDD3  D340102D  2C3B3868  01A80B4F  6E9E9A02  18B96ADD  8B162C41
69F5E787  700C2059  5BF765E6  10C5CB57  2C1BB16E
```

A.1.4.2 Représentation binaire

Pour faciliter la lecture de la représentation binaire des données, des lignes blanches séparent les groupes de champs logiquement associés (notamment les paires longueur/valeur); un saut de ligne sépare les champs; un espace sépare les caractères dans une chaîne de caractères; un point (.) indique les limites d'octets et un "x" représente un bit à zéro utilisé pour bourrer l'octet final à une limite d'octet.

1	Bit de mappage = 1 indique que "children" figure
0000010.0 1001010 .1101111 1.101000 11.01110	Longueur de name.givenName = 4 name.givenName = "John"
000.00001 101.0000	Longueur de name.initial = 1 name.initial = "P"
0000.0101 1010.011 11011.01 110100.1 1110100 .1101000	Longueur de name.familyName = 5 name.familyName = "Smith"
0.0000001 0.0110011	Longueur de (employee) number = 1 (employee) number = 51
0.0001000 1.000100 11.01001 111.0010 1100.101 11000.11 111010.0 1101111 .1110010	Longueur de title = 8 title = "Director"
0.0001000 0.110001 01.11001 011.0111 0110.001 01100.00 011100.1 0110001 .0111111	Longueur de dateOfHire = 8 dateOfHire = "19590717"
0.0000100 1.001101 11.00001 111.0010 1111.001	Longueur de nameOfSpouse.givenName = 4 nameOfSpouse.givenName = "Mary"
00000.001 10101.00	Longueur de nameOfSpouse.initial = 1 nameOfSpouse.initial = "T"
000001.01 101001.1 1101101 .1101001 1.110100 11.01000	Longueur de nameOfSpouse.familyName = 5 nameOfSpouse.familyName = "Smith"
000.00010	Nombre d'enfants
000.00101 101.0010 1100.001 11011.00 111000.0 1101000	Longueur de children[0].givenName = 5 children[0].givenName = "Ralph"
.00000001 .1010100	Longueur de children[0].initial = 1 children[0].initial = "T"
0.0000101 1.010011 11.01101 110.1001 1110.100 11010.00	Longueur de children[0].familyName = 5 children[0].familyName = "Smith"
000010.00 011000.1 0111001 .0110101 0.110111 01.10001 011.0001 0110.001 01100.01	Longueur de children[0].dateOfBirth = 8 children[0].dateOfBirth = "19571111"
000001.01 101001.1 1110101 .1110011 1.100001 11.01110	Longueur de children[1].givenName = 5 children[1].givenName = "Susan"

```
000.00001                                                    Longueur de children[1].initial = 1
100.0010                                                     children[1].initial = "B"

0000.0101                                                    Longueur de children[1].familyName = 5
1001.100  11011.11  110111.0  1100101  .1110011             children[1].familyName = "Jones"

0.0001000                                                    Longueur de children[1].dateOfBirth = 8
0.110001  01.11001  011.0101  0111.001  01100.00  011011.1  0110001  .0110111x children[1].dateOfBirth = "19590717"
```

A.2 Enregistrement utilisant des contraintes appliquées aux sous-types

Cet exemple est le même que celui donné à l'article A.1, sauf qu'il utilise la notation de sous-type pour imposer des contraintes à certains éléments.

A.2.1 Description ASN.1 de la structure d'enregistrement

La structure de l'enregistrement "PersonnelRecord" hypothétique est décrite formellement en utilisant la notation ASN.1 spécifiée dans la Rec. UIT-T X.680 | ISO/CEI 8824-1 pour la définition des types.

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET {
        name            Name,
        title           [0] VisibleString,
        number          EmployeeNumber,
        dateOfHire      [1] Date,
        nameOfSpouse    [2] Name,
        children        [3] IMPLICIT
            SEQUENCE OF ChildInformation  DEFAULT  {} }

ChildInformation ::= SET
        { name          Name,
          dateOfBirth   [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
      {givenName        NameString,
       initial          NameString (SIZE(1)),
       familyName       NameString}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString (FROM("0".."9") ^ SIZE(8)) -- YYYYMMDD

NameString ::= VisibleString (FROM("a".."z" | "A".."Z" | "-.") ^ SIZE(1..64))
```

A.2.2 Description ASN.1 d'une valeur d'enregistrement

La valeur de l'enregistrement PersonnelRecord "John Smith" est décrite formellement ci-dessous en utilisant la notation ASN.1.

```
{ name {givenName "John",initial "P",familyName "Smith"},
  title         "Director",
  number        51,
  dateOfHire            "19710917",
  nameOfSpouse          {givenName "Mary",initial "T",familyName "Smith"},
  children
      {{name {givenName "Ralph",initial "T",familyName "Smith"},
             dateOfBirth "19571111"},
       {name {givenName "Susan",initial "B",familyName "Jones"},
             dateOfBirth "19590717"}}}
```

A.2.3 Représentation ALIGNED PER de cette valeur d'enregistrement

La représentation de la valeur d'enregistrement donnée ci-dessus (après application de la variante ALIGNED des règles de codage compact définies dans la présente Recommandation | Norme internationale) est donnée ci-après. Le codage hexadécimal est suivi d'une description commentée du codage sous forme binaire. Dans la représentation binaire, les "x" représentent des bits de bourrage codés comme des bits zéro et utilisés pour caler un champ à l'octet.

La longueur de codage est de 74 octets. A titre de comparaison, la même valeur PersonnelRecord codée selon la variante UNALIGNED des règles PER est de 61 octets, dans la forme de longueur définie des règles BER elle est d'au moins 136 octets et dans la forme de longueur non définie des règles BER elle est d'au moins 161 octets.

A.2.3.1 Représentation hexadécimale

```
864A6F68 6E501053 6D697468 01330844 69726563 746F7219 7109170C 4D617279
5410536D 69746802 1052616C 70685410 536D6974 68195711 11105375 73616E42
104A6F6E 65731959 0717
```

A.2.3.2 Représentation binaire

Pour faciliter la lecture de la représentation binaire des données, des lignes blanches sont utilisées pour séparer les champs qui sont logiquement associés (normalement des couples longueur/valeur); un saut de ligne sépare les champs; un espace sépare les caractères dans une chaîne de caractères et les "x" représentent des bits de bourrage à zéro, parfois utilisés pour caler un champ à l'octet.

1	Bit de mappage = 1 indique que "children" figure
000011x	Longueur de name.givenName = 4
01001010 01101111 01101000 01101110	name.givenName = "John"
01010000	name.initial = "P"
000100xx	Longueur de name.familyName = 5
01010011 01101101 01101001 01110100 01101000	name.familyName = "Smith"
00000001	Longueur de (employee) number = 1
00110011	(employee) number = 51
00001000	Longueur de title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010	title = "Director"
0001 1001 0111 0001 0000 1001 0001 0111	dateOfHire = "19590717"
000011xx	Longueur de nameOfSpouse.givenName = 4
01001101 01100001 01110010 01111001	nameOfSpouse.givenName = "Mary"
01010100	nameOfSpouse.initial = "T"
000100xx	Longueur de nameOfSpouse.familyName = 5
01010011 01101101 01101001 01110100 01101000	nameOfSpouse.familyName = "Smith"
00000010	Nombre d'enfants
000100xx	Longueur de children[0].givenName = 5
01010010 01100001 01101100 01110000 01101000	children[0].givenName = "Ralph"
01010100	children[0].initial = "T"
000100xx	Longueur de children[0].familyName = 5
01010011 01101101 01101001 01110100 01101000	children[0].familyName = "Smith"
0001 1001 0101 0111 0001 0001 0001 0001	children[0].dateOfBirth = "19571111"
000100xx	Longueur de children[1].givenName = 5
01010011 01110101 01110011 01100001 01101110	children[1].givenName = "Susan"
01000010	children[1].initial = "B"
000100xx	Longueur de children[1].familyName = 5
01001010 01101111 01101110 01100101 01110011	children[1].familyName = "Jones"
0001 1001 0101 1001 0000 0111 0001 0111	children[1].dateOfBirth = "19590717"

A.2.4 Représentation UNALIGNED PER de cette valeur d'enregistrement

La représentation de la valeur d'enregistrement donnée ci-dessus (après application de la variante UNALIGNED des règles de codage compact définies dans la présente Recommandation | Norme internationale) est représentée ci-dessous. Le codage hexadécimal est suivi d'une description commentée de la forme de codage binaire. A noter qu'aucun bit de bourrage ne figure dans la variante UNALIGNED et que les caractères sont codés avec le plus petit nombre de bits possible.

La longueur de ce codage est de 61 octets. A titre de comparaison, la valeur de PersonnelRecord codée selon la variante ALIGNED des règles PER est de 74 octets, dans la forme de longueur définie des règles BER elle est d'au moins 136 octets et dans la forme de longueur non définie des règles BER elle est d'au moins 161 octets.

A.2.4.1 Représentation hexadécimale

```
865D51D2  888A5125  F1809984  44D3CB2E  3E9BF90C  B8848B86  7396E8A8  8A5125F1
81089B93  D71AA229  4497C632  AE222222  985CE521  885D54C1  70CAC838  B8
```

A.2.4.2 Représentation binaire

Pour faciliter la lecture de la représentation binaire des données, des lignes blanches séparent les groupes de champs logiquement associés (notamment les paires longueur/valeur); un saut de ligne sépare les champs; un espace sépare les caractères dans une chaîne de caractères; un point (.) indique les limites d'octets et un "x" représente un bit à zéro utilisé pour bourrer l'octet final à une limite d'octet.

1	Bit de mappage = 1 indique que "children" figure
000011 0.01011 101.010 10001.1 101001	Longueur de name.givenName = 4 name.givenName = "John"
0.10001	name.initial = "P"
000.100 01010.0 101000 1.00100 101.111 10001.1	Longueur de name.familyName = 5 name.familyName = "Smith"
0000000.1 0011001.1	Longueur de (employee) number = 1 (employee) number = 51
0000100.0 1000100 .1101001 1.110010 11.00101 110.0011 1110.100 11011.11 111001.0	Longueur de title = 8 title = "Director"
0001 100.1 0111 000.1 0000 100.1 0001 011.1	dateOfHire = "19590717"
000011 0.01110 011.100 10110.1 110100	Longueur de nameOfSpouse.givenName = 4 nameOfSpouse.givenName = "Mary"
0.10101	nameOfSpouse.initial = "T"
000.100 01010.0 101000 1.00100 101.111 10001.1	Longueur de nameOfSpouse.familyName = 5 nameOfSpouse.familyName = "Smith"
0000001.0	Nombre d'enfants
000100 0.10011 011.100 10011.1 101011 1.00011	Longueur de children[0].givenName = 5 children[0].givenName = "Ralph"
010.101	children[0].initial = "T"
00010.0 010100 1.01000 100.100 10111.1 100011 0.001 1001 0.101 0111 0.001 0001 0001 0001 0.001 1001 0.101 0111 0.001 0001 0.001 0001	Longueur de children[0].familyName = 5 children[0].familyName = "Smith" children[0].dateOfBirth = "19571111"
0.00100 010.100 11000.0 101110 0.11100 101.001	Longueur de children[1].givenName = 5 children[1].givenName = "Susan"
00001.1	children[1].initial = "B"
000100 0.01011 101.010 10100.1 100000 1.01110	Longueur de children[1].familyName = 5 children[1].familyName = "Jones"
000.1 1001 010.1 1001 000.0 0111 000.1 0111xxx	children[1].dateOfBirth = "19590717"

A.3 Enregistrement qui utilise des marqueurs d'extension

A.3.1 Description ASN.1 de la structure de l'enregistrement

La structure de l'enregistrement "PersonnelRecord" hypothétique est décrite de façon formelle ci-après en utilisant la notation ASN.1 spécifiée dans la Rec. UIT-T X.680 | ISO/CEI 8824-1 pour la définition des types non extensibles et la Rec. UIT-T X.680/Amd. 1 | ISO/CEI 8824-1/Amd. 1 pour la définition des types extensibles.

```
PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET {
        name        Name,
        title       [0] VisibleString,
        number      EmployeeNumber,
```

```
            dateOfHire      [1] Date,
            nameOfSpouse    [2] Name,
            children        [3] IMPLICIT
                 SEQUENCE (SIZE(2, ...)) OF ChildInformation  OPTIONAL  {},
        .. }

ChildInformation ::= SET
            { name          Name,
              dateOfBirth   [0] Date,
              ...,
              sex           [1] IMPLICIT ENUMERATED {male(1), female(2), unknown(3)} OPTIONAL
            }

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
            {givenName      NameString,
             initial        NameString (SIZE(1)),
             familyName     NameString,
             ...}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER (0..9999, ...)

Date ::= [APPLICATION 3] IMPLICIT VisibleString (FROM("0".."9") ^ SIZE(8, ..., 9..20)) -- YYYYMMDD

NameString ::= VisibleString (FROM("a".."z" | "A".."Z" | "-.") ^ SIZE(1..64, ...))
```

A.3.2 Description ASN.1 de la valeur d'enregistrement

La valeur d'enregistrement PersonnelRecord "John Smith" est décrite formellement ci-après en utilisant la notation ASN.1.

```
{ name {givenName "John",initial "P",familyName "Smith"},
  title         "Director",
  number        51,
  dateOfHire            "19710917",
  nameOfSpouse          {givenName "Mary",initial "T",familyName "Smith"},
  children
      {{name {givenName "Ralph",initial "T",familyName "Smith"},
             dateOfBirth "19571111"},
       {name {givenName "Susan",initial "B",familyName "Jones"},
             dateOfBirth "19590717", sex female}}}
```

A.3.3 Représentation ALIGNED PER de cette valeur d'enregistrement

La représentation de la valeur d'enregistrement donnée ci-dessus (après application de la variante ALIGNED des règles de codage compact définies dans la présente Recommandation | Norme internationale) est donnée ci-après. Le codage hexadécimal est suivi d'une description commentée du codage sous forme binaire. Dans la représentation binaire, les "x" représentent des bits de bourrage codés comme des bits zéro, parfois utilisés pour caler un champ à l'octet.

La longueur de codage est de 83 octets. A titre de comparaison, la même valeur PersonnelRecord codée selon la variante UNALIGNED des règles PER est de 65 octets, dans la forme de longueur définie des règles BER elle est d'au moins 139 octets et dans la forme de longueur non définie des règles BER elle est d'au moins 164 octets.

A.3.3.1 Représentation hexadécimale

```
40C04A6F  686E5008  536D6974  68000033  08446972  6563746F  72001971  0917034D
61727954  08536D69  74680100  52616C70  68540853  6D697468  00195711  11820053
7573616E  42084A6F  6E657300  19590717  010140
```

A.3.3.2 Représentation binaire

Pour faciliter la lecture de la représentation binaire des données, des lignes blanches sont utilisées pour séparer les champs qui sont logiquement associés (notamment les couples longueur/valeur); un saut de ligne sépare les champs; un espace sépare les caractères dans une chaîne de caractères et les "x" représentent des bits de bourrage à zéro, parfois utilisés pour caler des champs à l'octet.

```
0                                   Aucune valeur d'extension dans PersonnelRecord
1                                   Bit de mappage = 1 indique que "children" figure
```

0	Aucune valeur d'extension dans "name"
0	Longueur dans la plage de la racine d'extension
0000 11xxxxxx	Longueur de name.givenName = 4
01001010 01101111 01101000 01101110	name.givenName = "John"
01010000	name.initial = "P"
0	Longueur dans la plage de la racine d'extension
000100x	Longueur de name.familyName = 5
01010011 01101101 01101001 01110100 01101000	name.familyName = "Smith"
0xxxxxxx	Valeur dans la plage de la racine d'extension
00000000 00110011	(employee) number = 51
00001000	Longueur de title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010	title = "Director"
0xxxxxxx	Longueur dans la plage de la racine d'extension
0001 1001 0111 0001 0000 1001 0001 0111	dateOfHire = "19590717"
0	Aucune valeur d'extension dans nameOfSpouse
0	Longueur dans la plage de la racine d'extension
000011	Longueur de nameOfSpouse.givenName = 4
01001101 01100001 01110010 01111001	nameOfSpouse.givenName = "Mary"
01010100	nameOfSpouse.initial = "T"
0	Longueur dans la plage de la racine d'extension
000100x	Longueur de nameOfSpouse.familyName = 5
01010011 01101101 01101001 01110100 01101000	nameOfSpouse.familyName = "Smith"
0	Nombre de "children" dans la plage de la racine d'extension
0	Aucune valeur d'extension dans children[0]
0	Aucune valeur d'extension dans children[0].name
0	Longueur dans la plage de la racine d'extension
000100xx xxxx	Longueur de children[0].givenName = 5
01010010 01100001 01101100 01110000 01101000	children[0].givenName = "Ralph"
01010100	children[0].initial = "T"
0	Longueur dans la plage de la racine d'extension
000100x	Longueur de children[0].familyName = 5
01010011 01101101 01101001 01110100 01101000	children[0].familyName = "Smith"
0xxxxxxx	Longueur dans la plage de la racine d'extension
0001 1001 0101 0111 0001 0001 0001 0001	children[0].dateOfBirth = "19571111"
1	Valeurs d'extension dans children[1]
0	Aucune valeur d'extension dans children[0].name
0	Longueur dans la plage de la racine d'extension
00010 0xxxxxxx	Longueur de children[1].givenName = 5
01010011 01110101 01110011 01100001 01101110	children[1].givenName = "Susan"
01000010	children[1].initial = "B"

0	Longueur dans la plage de la racine d'extension
000100x	Longueur de children[1].familyName = 5
01001010 01101111 01101110 01100101 01110011	children[1].familyName = "Jones"
0xxxxxxx	Longueur dans la plage de la racine d'extension
0001 1001 0101 1001 0000 0111 0001 0111	children[1].dateOfBirth = "19590717"
0000000	Longueur de la table de mappage d'addition d'extension de children[1] = 1
1	Une valeur d'extension figure pour "sex"
00000001	Longueur du codage complet de "sex"
01xxxxxx	Codage complet de "sex" = female

A.3.4 Représentation UNALIGNED PER de cette valeur d'enregistrement

La représentation de la valeur d'enregistrement donnée ci-dessus (après application de la variante UNALIGNED des règles de codage compact définies dans la présente Recommandation | Norme internationale) est représentée ci-dessous. Le codage représenté en hexadécimal est suivi d'une description commentée de la forme de codage binaire. A noter qu'aucun bit de bourrage ne figure dans la variante UNALIGNED et que les caractères sont codés avec le plus petit nombre de bits possible.

La longueur de ce codage est de 65 octets. A titre de comparaison, la valeur de PersonnelRecord codée selon la variante ALIGNED des règles PER est de 83 octets, dans la forme de longueur définie des règles BER elle est d'au moins 139 octets et dans la forme de longueur non définie des règles BER elle est d'au moins 164 octets.

A.3.4.1 Représentation hexadécimale

```
40CBAA3A 5108A512 5F180330 889A7965 C7D37F20 CB8848B8 19CE5BA2 A114A24B
E3011372 7AE35422 94497C61 95711118 22985CE5 21842EAA 60B832B2 0E2E0202
80
```

A.3.4.2 Représentation binaire

Pour faciliter la lecture de la représentation binaire des données, des lignes blanches séparent les groupes de champs logiquement associés (notamment les paires longueur/valeur); un saut de ligne sépare les champs; un espace sépare les caractères dans une chaîne de caractères; un point (.) indique les limites d'octets et un "x" représente un bit à zéro utilisé pour bourrer l'octet final à une limite d'octet.

0	Aucune valeur d'extension dans PersonnelRecord
1	Bit de mappage = 1 indique que "children" figure
0	Aucune valeur d'extension dans le "name"
0	Longueur dans la plage de la racine d'extension
0000.11	Longueur de name.givenName = 4
001011 .101010 10.0011 1010.01	name.givenName = "John"
010001	name.initial = "P"
.0	Longueur dans la plage de la racine d'extension
000100	Longueur de name.familyName = 5
0.10100 101.000 10010.0 101111 1.00011	name.familyName = "Smith"
0	Valeur dans la plage de la racine d'extension
00.00000011.0011	(employee) number = 51
0000.1000	Longueur de title = 8
1000.100 11010.01 111001.0 1100101 1100011 1.110100 11.01111 111.0010	title = "Director"
0	Longueur dans la plage de la racine d'extension
000.1 1001 011.1 0001 000.0 1001 000.1 0111	dateOfHire = "19590717"

0	Aucune valeur d’extension dans nameOfSpouse
0	Longueur dans la plage de la racine d’extension
0.00011	Longueur de nameOfSpouse.givenName = 4
001.110 01110.0 101101 1.10100	nameOfSpouse.givenName = "Mary"
010.101	nameOfSpouse.initial = "T"
0	Longueur dans la plage de la racine d’extension
0001.00	Longueur de nameOfSpouse.familyName = 5
010100 .101000 10.0100 1011.11 100011	nameOfSpouse.familyName = "Smith"
.0	Nombre de "children" dans la plage de la racine d’extension
0	Aucune valeur d’extension dans children[0]
0	Aucune valeur d’extension dans children[0].name
0	Longueur dans la plage de la racine d’extension
0001.00	Longueur de children[0].givenName = 5
010011 .011100 10.0111 1010.11 100011	children[0].givenName = "Ralph"
.010101	children[0].initial = "T"
0	Longueur dans la plage de la racine d’extension
0.00100	Longueur de children[0].familyName = 5
010.100 10100.0 100100 1.01111 100.011	children[0].familyName = "Smith"
0	Longueur dans la plage de la racine d’extension
0001 .1001 0101 .0111 0001 .0001 0001 .0001	children[0].dateOfBirth = "19571111"
1	Valeurs d’extension dans children[1]
0	Aucune valeur d’extension dans children[0].name
0	Longueur dans la plage de la racine d’extension
0.00100	Longueur de children[1].givenName = 5
010.100 11000.0 101110 0.11100 101.001	children[1].givenName = "Susan"
00001.1	children[1].initial = "B"
0	Longueur dans la plage de la racine d’extension
000100	Longueur de children[1].familyName = 5
.001011 10.1010 1010.01 100000 .101110	children[1].familyName = "Jones"
0	Longueur dans la plage de la racine d’extension
0.001 1001 0.101 1001 0.000 0111 0.001 0111	children[1].dateOfBirth = "19590717"
0.000000	Longueur de la table de mappage d’ajouts d’extensions de children[1] = 1
1	Une valeur d’extension figure pour "sex"
0.0000001	Longueur du codage complet de "sex"
0.1xxxxxx	Codage complet de "sex" = female
x	Bit de bourrage pour créer le codage complet de PersonnelRecord

Annexe B

Observations sur la combinaison de contraintes visibles par les règles PER

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Certaines propriétés peuvent être observées lorsque des éléments de sous-type, dont chacun peut être séparément visible par les règles PER, sont combinés. Ces propriétés sont décrites ci-après au moyen d'exemples:

B.1 La contrainte de taille effective pour:

```
A ::= IA5String (SIZE(1..4) | SIZE(9..10))
```

est

```
A ::= IA5String (SIZE(1..4 | 9..10))
```

B.2 Lorsqu'une contrainte PermittedAlphabet (alphabet permis) est combinée dans la même spécification de sous-type avec d'autres contraintes PermittedAlphabet, aucune contrainte PermittedAlphabet contenant moins de caractères que tous les caractères du type non contraint n'est effective, sauf s'il existe une spécification PermittedAlphabet qui est un surensemble de toutes les autres spécifications PermittedAlphabet dans cette spécification de sous-type. En outre, si des contraintes de taille sont incluses dans cette spécification de contrainte, la contrainte effective de taille du PermittedAlphabet qui est un surensemble, doit être composée avec chacune des spécifications de contrainte par l'opération INTERSECTION, en sorte que la contrainte de taille effective soit un surensemble de toutes les autres contraintes de taille imposées au type. Par exemple:

```
B ::= IA5String ( FROM ("AB") ^ SIZE(1..2) |
                  FROM ("DE") ^ SIZE(3) |
                  FROM ("ABCDE") ^ (SIZE(1..5) )
```

à une contrainte de taille effective et une contrainte PermittedAlphabet effective de:

```
B ::= IA5String (FROM ("ABCDE") ^ SIZE(1..5))
```

car il s'agit d'un surensemble de l'expression plus complexe ci-dessus – qui est donc visible par les règles PER. Par ailleurs, dans l'exemple suivant, la contrainte PermittedAlphabet effective est le jeu complet de caractères permis pour IA5String, puisque aucune contrainte PermittedAlphabet équivalente unique ne peut être écrite. La contrainte sur la production suivante n'est donc pas visible par les règles PER:

```
C ::= IA5String (FROM("AB") | FROM("CD"))        -- Ce qui n'est pas équivalent à (FROM("ABCD"))
```

B.3 Les contraintes de taille peuvent être librement combinées à condition qu'aucune contrainte PermittedAlphabet ne soit impliquée. Par exemple,

```
E ::= IA5String (SIZE(1..4) | SIZE(5..10) ^ FROM("ABCD") | SIZE(6..10))
```

ne met en jeu aucune contrainte de longueur visible par les règles PER (puisque la longueur 5 ne peut pas se présenter pour tous les caractères possibles), alors que si nous avions

```
E ::= IA5String (SIZE(1..4) | SIZE(6..10) ^ FROM("ABCD") | SIZE(6..10))
```

une contrainte de longueur visible par les règles PER s'appliquerait à SIZE(1..4 | 6..10), bien que la contrainte PermittedAlphabet effective soit le jeu de tous les caractères IA5String. Dans ce cas, FROM("ABCD") ne constitue pas une contrainte visible par les règles PER, car elle ne s'applique pas à toutes les valeurs possibles de E (par exemple, si la longueur de chaîne est 1, alors le caractère n'est pas restreint à l'un des caractères "ABCD").

Annexe C

Prise en charge des algorithmes PER

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Une norme d'application ou un profil normalisé international, peut spécifier le type de règles de codage compact qu'il y a lieu de prendre en charge ainsi que les syntaxes de transfert correspondantes qui devront être proposées ou acceptées lors de la négociation.

Lorsque ces règles régissent l'emploi de codages à compatibilité assurée et/ou de codages canoniques dans le cadre de types EMBEDDED PDV (ou EXTERNAL) ou CHARACTER STRING, cela doit être clairement indiqué.

On trouvera ci-après quelques directives pouvant servir à la production d'un texte normatif.

C.1 Un codage canonique est normalement utilisé lorsqu'on applique des fonctions de sécurité au codage (voir l'Annexe D de la Rec. UIT-T X.690 | ISO/CEI 8825-1). L'utilisation de l'algorithme CANONICAL-PER peut entraîner une notable augmentation des coûts d'utilisation de l'unité centrale lorsque la valeur à coder comporte un type ensemble-de. Il n'est donc pas recommandé d'utiliser souvent cet algorithme pour les protocoles si des fonctions de sécurité ne sont pas requises.

C.2 Lorsqu'une valeur de syntaxe abstraite contient des éléments imbriqués qui sont codés à l'aide d'une syntaxe de transfert ou d'une syntaxe abstraite différente de celle qui est associée à la valeur de syntaxe abstraite, il est vivement recommandé que ces éléments soient codés en compatibilité assurée, avec application d'une règle de codage canonique si les aspects sécurité sont importants. Dans ce contexte, il convient de prêter une attention particulière au niveau d'application des règles de l'ISO/CEI 10646-1, pour les types BMPString ou UniversalString, car le codage canonique n'est garanti que pour le niveau de réalisation 1 de l'ISO/CEI 10646-1.

C.3 Lorsqu'un contexte de présentation est établi pour un seul sens du flux de données, il est vivement recommandé de coder les données en compatibilité assurée.

C.4 Lorsqu'un contexte de présentation est établi pour le flux de données dans les deux sens, l'emploi de l'algorithme BASIC-PER peut apporter de notables avantages en termes de flexibilité et d'économie.

C.5 Il est vivement recommandé que toutes les mises en œuvre, supportant le décodage de toute syntaxe de transfert en variante PER ALIGNED, puissent supporter le décodage de la variante BASIC-PER ALIGNED (et donc de la variante CANONICAL-PER ALIGNED) et puissent accepter des contextes de présentation identifiés au moyen d'une quelconque de ces deux règles de codage, à condition que les contextes soient établis pour la réception des données par cette mise en œuvre. La même recommandation est applicable à la variante UNALIGNED.

C.6 A des fins d'interfonctionnement, il est recommandé que toutes les mises en œuvre des règles PER supportent aussi bien la variante ALIGNED que la variante UNALIGNED (le supplément de complexité correspondant étant petit). Il relève également d'une décision de gestion locale de savoir quelle variante sera offerte dans une instance de communication (une des deux ou les deux) et quelle variante sera acceptée si les deux sont offertes. Si une seule est offerte, il y aura lieu qu'elle soit acceptée.

C.7 L'acceptation des présentes recommandations est particulièrement importante pour les fournisseurs d'outils universels. Lorsqu'une mise en œuvre est spécifique à une application particulière, on peut considérer comme tout à fait acceptable qu'elle ne prenne en charge qu'une seule syntaxe de transfert compatible avec les règles PER (éventuellement spécifiée par le concepteur de cette application).

Annexe D

Prise en charge des règles d'extensibilité ASN.1

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

D.1 Les règles de codage compact PER dépendent de la définition complète du type auquel elles s'appliquent. En général, si des modifications autres que celles de nature purement syntaxique sont apportées à la définition du type, le codage de toutes les valeurs utilisant la partie de la spécification modifiée sera affecté. En particulier, l'ajout de composants optionnels à une séquence, la conversion d'un composant en un CHOICE de ce composant ou en un autre type, la suppression, la diminution ou le renforcement de contraintes sur un composant, risquent de modifier le codage des valeurs du type.

D.2 Néanmoins, les règles de codage PER ont été conçues de façon à garantir que les prescriptions énoncées pour les règles de codage dans le modèle ASN.1 d'extension de type (voir Rec. UIT-T X.680/Amd. 1 | ISO/CEI 8824-1/Amd. 1) soient satisfaites.

D.3 Lorsqu'un type ne fait pas partie d'une séquence d'extension (aucun marqueur d'extension ne figure), le texte figurant plus haut dans la présente annexe s'applique: les règles PER ne permettent pas la prise en charge de l'extensibilité de ce type. Lorsqu'un type séquence ou ensemble comporte un marqueur d'extension, mais ne comporte pas d'ajouts d'extensions, son codage ne nécessite un bit supplémentaire (qui peut devenir un octet supplémentaire, du fait d'un bourrage dans les variantes ALIGNED) qu'avec le même type sans marqueur d'extension. Lorsque des ajouts figurent dans le type et sont effectivement transmis à une instance de communication, le codage demande encore environ un octet supplémentaire, plus un champ de longueur additionnel pour chaque ajout d'extension qui est transmis, que le même type où l'on aurait supprimé le marqueur d'extension.

D.4 Il importe de noter que l'ajout, ainsi que la suppression d'un marqueur d'extension, modifient les bits transmis sur la ligne et nécessitent en général la modification du numéro de version du protocole.

D.5 Ni l'inclusion d'un marqueur d'extension dans un ensemble d'objets informationnels, ni l'ajout ou la suppression de spécifications d'exceptions, n'entraînent de modification de codage, mais ces opérations peuvent évidemment se traduire par des changements du comportement requis d'une mise en œuvre, et donc nécessiter également une modification du numéro de version du protocole.

Annexe E

Complément didactique sur la concaténation de codages conformes aux règles PER

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

E.1 Les codages PER délimitent automatiquement la connaissance à avoir des règles et du type de codage. Les codages complets, dans les variantes ALIGNED et UNALIGNED, sont toujours des multiples de 8 bits.

E.2 En vue d'intégrer les codages PER dans le protocole OSI de la couche présentation, les codages selon les variantes ALIGNED et UNALIGNED peuvent être concaténés dans l'option chaîne d'octets.

Annexe F

Affectation de valeurs à un identificateur d'objet

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Les valeurs suivantes, de types identificateur d'objet et descripteur d'objet, sont affectées dans la présente Recommandation | Norme internationale:

Pour l'algorithme BASIC-PER, variante ALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) aligned (0)}

"Codage compact d'un unique type ASN.1 (règles de base et champs calés à l'octet)"

Pour l'algorithme BASIC-PER, variante UNALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) unaligned (1)}

"Codage compact d'un unique type ASN.1 (règles de base et champs non calés à l'octet)"

Pour l'algorithme CANONICAL-PER, variante ALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) aligned (0)}

"Codage compact d'un unique type ASN.1 (règles canoniques et champs calés à l'octet)"

Pour l'algorithme CANONICAL-PER, variante UNALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) unaligned (1)}

"Codage compact d'un unique type ASN.1 (règles canoniques et champs non calés à l'octet)"