

Remplacée par une version plus récente



UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

X.680

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(07/94)

**RÉSEAUX POUR DONNÉES ET INTERCONNEXION
DES SYSTÈMES OUVERTS**

**RÉSEAUTAGE OSI ET ASPECTS DES SYSTÈMES
– NOTATION DE SYNTAXE ABSTRAITE
NUMÉRO UN**

**TECHNOLOGIES DE L'INFORMATION –
NOTATION DE SYNTAXE ABSTRAITE
NUMÉRO UN: SPÉCIFICATION DE LA
NOTATION DE BASE**

Recommandation UIT-T X.680

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 n° 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.680 de l'UIT-T a été approuvé le 1^{er} juillet 1994. Son texte est publié, sous forme identique, comme Norme internationale ISO/CEI 8824-1.

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.

© UIT 1996

Droits de reproduction réservés. Aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'UIT.

Remplacée par une version plus récente

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX POUR DONNÉES ET INTERCONNEXION DES 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 communication	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

	<i>Page</i>
Résumé	iv
Introduction	v
1 Domaine d'application.....	1
2 Références normatives	1
2.1 Recommandations Normes internationales identiques.....	1
2.2 Autres références	2
3 Définitions.....	2
3.1 Spécification des objets informationnels	2
3.2 Spécification des contraintes.....	2
3.3 Paramétrisation des spécifications ASN.1	3
3.4 Définition du service de présentation.....	3
3.5 Spécification du protocole de présentation	3
3.6 Structure pour l'identification des organisations	3
3.7 Jeu de caractères universels codés sur multi-octets (UCS).....	3
3.8 Définitions supplémentaires.....	3
4 Abréviations	7
5 Notation.....	7
5.1 Productions	8
5.2 Collections au choix d'une production	8
5.3 Exemple de production	8
5.4 Mise en page	8
5.5 Récursivité	9
5.6 Pointage d'une collection de séquences	9
5.7 Pointage d'un item.....	9
5.8 Notations abrégées	9
6 Étiquettes.....	10
7 Utilisation de la notation ASN.1	11
8 Jeu de caractères ASN.1	11
9 Items ASN.1	12
9.1 Règles générales.....	12
9.2 Référence de type.....	12
9.3 Identificateur	12
9.4 Référence de valeur.....	12
9.5 Référence de module.....	12
9.6 Commentaire.....	13
9.7 Item vide	13
9.8 Item numéro	13
9.9 Item chaîne binaire.....	13
9.10 Item chaîne hexadécimale.....	13
9.11 Item chaîne de caractères	14
9.12 Item affectation	14
9.13 Séparateur de plage	14
9.14 Points de suspension	15
9.15 Items à caractère unique.....	15
9.16 Items mots réservés.....	15

Remplacée par une version plus récente

Page

10	Définition de module.....	16
11	Référenciation des définitions de types et de valeurs.....	19
12	Notation de prise en charge des références à des composantes ASN.1.....	20
13	Affectation de types et de valeurs	21
14	Définition des types et valeurs	22
15	Notation du type booléen	24
16	Notation du type entier (Integer).....	24
17	Notation du type énuméré	25
18	Notation du type réel.....	26
19	Notation du type chaîne binaire (bitstring).....	27
20	Notation du type chaîne d'octets (octetstring)	28
21	Notation du type néant (Null).....	28
22	Notation des types séquence	29
23	Notation des types séquence-de	30
24	Notation des types ensemble	31
25	Notation des types ensemble-de.....	31
26	Notation des types choix	32
27	Notation des types sélection.....	33
28	Notation des types étiquetés.....	33
29	Notation du type identificateur d'objet	34
30	Notation du type pdv encapsulé	36
31	Notation du type externe	38
32	Les types chaînes de caractères	40
33	Notation des types chaîne de caractères	40
34	Définition des types chaînes de caractères restreintes.....	40
35	Dénomination des caractères et collections de caractères définis dans ISO/CEI 10646-1.....	44
	35.1 Spécification du module ASN1-CHARACTER-MODULE.....	44
36	Ordre canonique des caractères.....	47
37	Définition des types chaînes de caractères non restreintes.....	49
38	Notation des types définis dans les articles 39 à 41	50
39	Temps généralisé.....	50
40	Temps universel	51
41	Type descripteur d'objets.....	52
42	Types contraints	52
43	Identificateur d'exception	53
44	Spécification d'un ensemble d'éléments	54
45	Éléments de sous-type.....	55
	45.1 Généralités	55
	45.2 Valeur unique.....	55
	45.3 Sous-type contenu	55
	45.4 Intervalle de valeurs.....	55
	45.5 Contrainte de taille.....	57
	45.6 Contrainte de type.....	57
	45.7 Alphabet autorisé	57
	45.8 Sous-typage interne.....	57

Remplacée par une version plus récente

Page

Annexe A – Utilisation de la notation ASN.1-88/90	60
A.1 Maintenance	60
A.2 Panachage de l'ASN.1-88/90 et de la notation ASN.1 actuelle	60
A.3 Migration vers la notation ASN.1 actuelle	60
Annexe B – Affectation par l'ISO de valeurs de composantes d'identificateur d'objet	63
Annexe C – Affectation par l'UIT-T de valeurs de composantes d'identificateur d'objet	64
Annexe D – Affectation commune de valeurs de composantes d'identificateur d'objet	65
Annexe E – Affectation de valeurs d'identificateurs d'objets	66
Annexe F – Exemples et conseils stylistiques	67
F.1 Exemple d'un enregistrement "salarié"	67
F.2 Directives pour l'utilisation de la notation	68
F.3 Identification des syntaxes abstraites	77
F.4 Sous-types	78
Annexe G – Annexe didactique sur les chaînes de caractères ASN.1	81
G.1 Prise en charge des chaînes de caractères en notation ASN.1	81
G.2 Les types chaîne universelle "UniversalString" et table multilingue "BMPString"	81
G.3 A propos des prescriptions de conformité à ISO/CEI 10646-1	82
G.4 Recommandations aux utilisateurs ASN.1 à propos de la conformité à ISO/CEI 10646-1	82
G.5 Sous-jeux adoptés comme paramètres de la syntaxe abstraite	83
G.6 Le type chaîne de caractères CHARACTER STRING	83
Annexe H – Caractéristiques remplacées	85
H.1 Utilisation des identificateurs devenus obligatoires	85
H.2 Valeur du type choix	85
H.3 Type ANY	85
H.4 Capacité de macro-notation	86
Annexe I – Notation du type (quelconque)	87
I.1 Notation du type	87
Annexe J – Les macro-notations	88
J.1 Introduction	88
J.2 Extensions aux items et au jeu de caractères ASN.1	88
J.3 Notation de macro-définition	90
J.4 Utilisation de la nouvelle notation	93
Annexe K – Récapitulatif de la notation ASN.1	94

Remplacée par une version plus récente

Résumé

La présente Recommandation | Norme internationale fournit une notation dite notation de syntaxe abstraite numéro un (ASN.1) pour la définition de la syntaxe des données informationnelles. Elle définit un certain nombre de types de données simples et spécifie une notation pour y faire référence et en spécifier les valeurs.

La notation ASN.1 peut être utilisée chaque fois que cela est nécessaire pour définir la syntaxe abstraite d'informations, sans que cela impose une contrainte quelconque sur la manière de coder cette information en transmission. Cette notation s'applique notamment mais pas exclusivement aux protocoles de la couche application.

Remplacée par une version plus récente

Introduction

La présente Recommandation | Norme internationale présente une notation normalisée pour la définition des types de données et de leurs valeurs. Un *type de donnée* (en abrégé, un *type*) est une classe informationnelle (une information numérique, textuelle, iconographique ou vidéo par exemple). Une *valeur de donnée* (en abrégé une *valeur*) est une instance d'une telle classe. La présente Recommandation | Norme internationale définit plusieurs types de base et les valeurs qui leur correspondent, ainsi que les règles pour les combiner en types et valeurs plus complexes.

Bien que cette notation normalisée soit définie dans le cadre général de l'OSI, elle peut servir à de nombreuses autres fins. Dans les couches inférieures du Modèle de référence de base de l'OSI (Rec. UIT-T X.200 | ISO/CEI 7498) et dans de nombreuses autres architectures de protocoles, chaque message est spécifié comme la valeur binaire d'une séquence d'octets. Dans la couche présentation de l'OSI (voir la Rec. UIT-T X.216 | ISO/CEI 8822), la nature des paramètres données d'utilisateur est différente. Toutefois, les spécifications de la couche application imposent la définition de types de données relativement complexes pour véhiculer leurs messages, indépendamment de leur représentation binaire. Pour spécifier les types de données, ces Recommandations nécessitent une notation qui ne détermine pas nécessairement la représentation de chaque valeur. Une telle notation doit être complétée par la spécification d'un ou plusieurs algorithmes appelés **règles de codage** qui déterminent les valeurs des octets des couches inférieures véhiculant les données de la couche application (appelées **syntaxes de transfert**). Le protocole de la couche présentation de l'OSI (voir la Rec. UIT-T X.226 | ISO/CEI 8823) peut négocier les syntaxes de transfert (**codages**) à utiliser.

Hors du contexte de l'OSI, l'utilité de la notion de valeur abstraite d'une classe quelconque (par exemple une image donnée à 256 couleurs) détachée de toute considération de codage particulier est de plus en plus reconnue, ce qui, pour interpréter correctement la représentation binaire de la valeur, nécessiterait de connaître (en général d'après le contexte) le type (classe) de la valeur représentée, ainsi que le mécanisme de codage utilisé. L'identification d'un type constitue donc une partie importante de la présente Recommandation | Norme internationale.

Une technique très générale pour définir un type compliqué au niveau abstrait consiste à définir un petit nombre de **types simples** en définissant toutes leurs valeurs possibles, puis de combiner ces types simples de diverses façons. A titre d'exemple, on peut citer les façons suivantes de définir de nouveaux types:

- a) étant donné une liste (ordonnée) de types existants, une valeur peut être formée comme une séquence (ordonnée) de valeurs, en prenant une valeur de chacun des types existants; la collection de toutes les valeurs possibles ainsi obtenues forme un nouveau type (si les types de la liste sont tous distincts, ce mécanisme peut être étendu pour permettre l'omission de certaines valeurs de la liste);
- b) étant donné un ensemble non ordonné de types (distincts) existants, une valeur peut être formée comme un ensemble (non ordonné) de valeurs, en prenant une valeur de chacun des types existants; la collection de tous les ensembles non ordonnés possibles ainsi obtenus forme un nouveau type; (là encore, le mécanisme peut être étendu pour permettre l'omission de certaines valeurs);
- c) étant donné un type simple existant, une valeur peut être formée comme une liste (ordonnée) ou un ensemble (non ordonné) de zéro, une ou plusieurs valeurs du type; la collection de tous les ensembles ou listes possibles ainsi obtenus forme un nouveau type;
- d) étant donné une liste de types (distincts), une valeur peut être choisie dans l'un quelconque de ces types; l'ensemble de toutes les valeurs possibles ainsi obtenues forme un nouveau type;
- e) étant donné un type, un nouveau type peut être induit comme un sous-ensemble de ce type, en appliquant à ses valeurs une contrainte structurelle ou une relation d'ordre quelconque.

Un aspect important d'une telle combinaison des types est que les règles de codage doivent permettre de reconnaître les différentes structures combinantes, assurant ainsi un codage non ambigu de la collection de valeurs des types de base. Ainsi, une **étiquette** est affectée à chaque type défini au moyen de la notation spécifiée dans la présente Recommandation | Norme internationale pour en permettre le codage non ambigu des valeurs.

Quatre classes d'étiquettes sont spécifiées dans la notation.

La première est la classe **universelle**. Les étiquettes de la classe universelle ne sont utilisées que selon les spécifications de la présente Recommandation | Norme internationale, chaque étiquette étant affectée:

- a) soit à un type unique;
- b) soit à un mécanisme de structuration.

Les utilisateurs de la présente notation ne sont pas autorisés à spécifier explicitement des étiquettes de la classe universelle dans leurs déclarations ASN.1, ces étiquettes étant prédéfinies et ne pouvant être explicitement spécifiées que dans la présente Recommandation | Norme internationale.

Remplacée par une version plus récente

Les trois autres classes d'étiquette sont la classe **application**, la classe **privée** et la classe **propre au contexte**. Aucune différence d'utilisation formelle n'existe entre ces trois classes. Là où une étiquette de la classe application est utilisée, il est généralement possible d'utiliser à la place une étiquette de la classe privée ou de la classe propre au contexte, au choix de l'utilisateur. La présence des trois classes est en grande partie due à des raisons historiques, mais F.2.12 fournit des indications sur la manière dont ces différentes classes sont généralement utilisées.

Les étiquettes sont principalement destinées au traitement machine et ne sont pas essentielles à la forme de notation lisible par l'homme, définie dans la présente Recommandation | Norme internationale. Toutefois, la nécessité de distinguer certains types se traduit en leur imposant d'avoir des étiquettes distinctes. L'affectation des étiquettes constitue donc un aspect important de l'utilisation de la présente notation.

NOTE – Dans la présente Recommandation | Norme internationale, des valeurs d'étiquette sont affectées à tous les types simples et mécanismes de structuration. Les restrictions imposées à l'utilisation de la notation garantissent de pouvoir utiliser les étiquettes en transfert pour identifier les valeurs de façon non ambiguë.

Les articles 8 à 29 inclus définissent les types simples pris en charge par la notation ASN.1 et spécifient la notation à utiliser pour faire référence à des types simples et pour définir de nouveaux types au moyen de ces types simples. Ils spécifient également la notation à utiliser pour spécifier les valeurs appartenant à des types définis en ASN.1.

Les articles 30 à 31 inclus définissent les types pris en charge par la notation ASN.1 pour véhiculer le codage complet des types ASN.1.

Les articles 32 à 37 inclus définissent les types de chaîne de caractères.

Les articles 38 à 41 inclus définissent certains types considérés comme étant d'utilité générale mais qui ne nécessitent aucune règle de codage supplémentaire.

Les articles 42 et 45 définissent une notation qui permet d'obtenir des sous-types à partir des valeurs d'un type parent.

L'Annexe A, qui fait partie intégrante de la présente Recommandation | Norme internationale, donne des indications sur la manière dont les utilisateurs de la présente Recommandation | Norme internationale peuvent faire référence aux types et valeurs ASN.1 définis au moyen de la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990.

L'Annexe B, qui fait partie intégrante de la présente Recommandation | Norme internationale, définit l'arbre des identificateurs d'objet pour les organismes relevant de l'ISO.

L'Annexe C, qui fait partie intégrante de la présente Recommandation | Norme internationale, définit l'arbre des identificateurs d'objets pour les organismes relevant de l'UIT-T.

L'Annexe D, qui fait partie intégrante de la présente Recommandation | Norme internationale, définit l'arbre des identificateurs d'objets utilisés conjointement par l'ISO et l'UIT-T.

L'Annexe E, qui fait partie intégrante de la présente Recommandation | Norme internationale, récapitule les valeurs d'identificateurs d'objets et de descripteurs d'objets affectées dans le cadre de la présente Recommandation | Norme internationale.

L'annexe F, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, fournit des exemples et des indications relatifs à l'utilisation de la notation ASN.1

L'Annexe G, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, est un exposé didactique sur les chaînes de caractères ASN.1.

L'Annexe H, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, décrit les caractéristiques de la version précédente de l'ASN.1 aujourd'hui obsolètes.

L'Annexe I, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, détaille la notation du type ANY (quelconque) aujourd'hui obsolète.

L'Annexe J, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, détaille les macro-notations aujourd'hui obsolètes.

L'Annexe K, qui ne fait pas partie intégrante de la présente Recommandation | Norme internationale, fournit un résumé de l'ASN.1 en utilisant la notation de l'article 5.

NORME INTERNATIONALE

RECOMMANDATION UIT-T

**TECHNOLOGIES DE L'INFORMATION –
NOTATION DE SYNTAXE ABSTRAITE NUMÉRO UN:
SPÉCIFICATION DE LA NOTATION DE BASE**

1 Domaine d'application

La présente Recommandation | Norme internationale spécifie une notation normalisée appelée notation de syntaxe abstraite numéro un (ASN.1) servant à définir les types de données, les valeurs et les contraintes imposées à ces types.

La présente Recommandation | Norme internationale:

- définit un certain nombre de types simples avec leurs étiquettes, et spécifie une notation pour la désignation de ces types et la spécification de leurs valeurs;
- définit des mécanismes pour construire de nouveaux types à partir de types plus élémentaires, et spécifie une notation pour définir de tels types, leur affecter des étiquettes, et en spécifier les valeurs;
- définit (par référence à d'autres Recommandations | Normes internationales) les jeux de caractères à utiliser en notation ASN.1;
- définit un certain nombre de types utiles (en utilisant la notation ASN.1) auxquels l'utilisateur de l'ASN.1 peut faire référence.

La notation ASN.1 peut être utilisée chaque fois qu'il est nécessaire de définir la syntaxe abstraite d'informations. Elle est en particulier applicable, mais non exclusivement, aux protocoles d'application.

Il est fait référence à la notation ASN.1 dans d'autres normes qui définissent les règles de codage pour des types ASN.1.

2 Références normatives

Les Recommandations et les Normes internationales suivantes contiennent des dispositions qui, par suite de la référence qui y est faite dans le présent document, 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 actuellement en vigueur.

2.1 Recommandations | Normes internationales identiques

- Recommandation UIT-T X.200 (1994) | ISO/CEI 7498-1:1994 *Technologie de l'information – Interconnexion des systèmes ouverts – Le modèle de référence de base: Modèle de référence de base.*
- Recommandation UIT-T X.216 (1994) | ISO/CEI 8822:1994 *Technologie de l'information – Interconnexion des systèmes ouverts – Définition du service de présentation.*
- Recommandation UIT-T X.226 (1994) | ISO/CEI 8823-1:1994 *Technologie de l'information – Interconnexion des systèmes ouverts – Protocole de présentation en mode connexion: Spécification du protocole.*
- 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.682 (1994) | ISO/CEI 8824-3:1995 *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN.1): 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.*
- Recommandation UIT-T X.691 (1995) | ISO/CEI 8825-2:1995 *Technologies de l'information – Règles de codage de la notation de syntaxe abstraite numéro un – Spécification des règles de codage en paquet.*

2.2 Autres références

- ISO *Registre international des jeux de caractères codés à utiliser avec les séquences d'échappement.*
- ISO/CEI 646:1991, *Technologies de l'information – Jeux ISO de caractères codés à 7 éléments pour l'échange d'informations.*
- ISO/CEI 2022:1994, *Technologies de l'information – Structure de code de caractères et techniques d'extension.*
- ISO 3166:1993, *Codes pour la représentation des noms de pays.*
- ISO 6523:1984, *Echange de données – Structures pour l'identification des organisations.*
- ISO 8601:1988, *Eléments de données et formats d'échange – Echange d'information – Représentation de la date et de l'heure.*
- ISO/CEI 10646-1:1993, *Technologies de l'information – Jeu universel de caractères à plusieurs octets Partie 1: Architecture et table multilingue.*
- Recommandation X.121 du CCITT (1992) *Plan de numérotage international pour les réseaux publics pour données.*
- Recommandation X.208 du CCITT (1988), *Spécification de la syntaxe abstraite numéro un (ASN.1).*
- ISO/CEI 8824:1990, *Technologies de l'information – Interconnexion de systèmes ouverts – Spécification de la notation de syntaxe abstraite numéro 1 (ASN.1).*

3 Définitions

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

3.1 Spécification des objets informationnels

La présente Recommandation | Norme internationale utilise les termes suivants, définis dans la Rec. UIT-T X.681 | ISO/CEI 8824-2:

- a) objet informationnel;
- b) classe d'objets informationnels;
- c) ensemble d'objets informationnels;
- d) type instance-de;
- e) type champ de classe d'objets.

3.2 Spécification des contraintes

La présente Recommandation | Norme internationale utilise les termes suivants, définis dans la Rec. UIT-T X.682 | ISO/CEI 8824-3:

- a) contrainte relationnelle de composante;
- b) contrainte tabulaire.

3.3 Paramétrisation des spécifications ASN.1

La présente Recommandation | Norme internationale utilise les termes suivants, définis dans la Rec. UIT-T X.683 | ISO/CEI 8824-4:

- a) type paramétré;
- b) valeur paramétrée.

3.4 Définition du service de présentation

La présente Recommandation | Norme internationale utilise les termes suivants, définis dans la Rec. UIT-T X.216 | ISO/CEI 8822:

- a) syntaxe abstraite;
- b) nom de syntaxe abstraite;
- c) ensemble contextuel défini;
- d) valeur de donnée de présentation;
- e) (une) syntaxe de transfert;
- f) nom de syntaxe de transfert.

3.5 Spécification du protocole de présentation

La présente Recommandation | Norme internationale utilise le terme suivant, défini dans la Rec. UIT-T X.226 | ISO/CEI 8823:

- identificateur de contexte de présentation

3.6 Structure pour l'identification des organisations

La présente Recommandation | Norme internationale utilise les termes suivants, définis dans ISO 6523:

- a) organisme émetteur;
- b) code d'organisation;
- c) prescripteur de code international (ICD).

3.7 Jeu de caractères universels codés sur multi-octets (UCS)

La présente Recommandation | Norme internationale utilise les termes suivants, définis dans ISO/CEI 10646-1:

- a) table multilingue (BMP) (*basic multilingual plane*);
- b) cellule;
- c) caractère de combinaison;
- d) symbole graphique;
- e) groupe;
- f) sous-ensemble limité;
- g) plan;
- h) rangée;
- i) sous-ensemble sélectionné.

3.8 Définitions supplémentaires

3.8.1 caractère abstrait: Ensemble de l'information associée à une cellule dans une table définissant un répertoire de caractères:

NOTE – Cette information comprend normalement tout ou partie des éléments suivants:

- a) un symbole graphique,
- b) le nom du caractère,
- c) la définition des fonctions associées au caractère lorsqu'il est utilisé dans des environnements particuliers.

3.8.2 valeur abstraite: Valeur dont la définition est basée uniquement sur le type, indépendamment de la manière dont elle est représentée par une règle de codage quelconque.

NOTE – L'utilisation de l'expression «valeur abstraite» sous-entend souvent que l'entité décrite varie probablement selon les règles de codage utilisées.

3.8.3 jeu de caractères ASN.1: Jeu de caractères spécifié à l'article 8 et utilisé en notation ASN.1.

3.8.4 spécification ASN.1: Collection d'un ou plusieurs modules ASN.1.

3.8.5 type associé: Type utilisé seulement pour définir la valeur et la notation de sous-type d'un type donné.

NOTE – Des types associés sont définis dans la présente Recommandation | Norme internationale lorsqu'il est nécessaire de bien indiquer qu'il existe une différence significative entre la façon dont le type est défini en ASN.1 et la façon de le coder. Les types associés n'apparaissent pas dans les spécifications d'utilisateurs.

3.8.6 type chaîne binaire (bitstring): Type simple dont chaque valeur distinctive est une séquence ordonnée de zéro, un ou plusieurs bits.

NOTE – Lorsqu'il est nécessaire de véhiculer des codages encapsulés d'une valeur abstraite, l'utilisation de valeurs de données de présentation (pdv) encapsulées constituera généralement un mécanisme plus souple que le type chaîne binaire pour annoncer la nature des codages ou s'accorder dessus.

3.8.7 type booléen: Type simple ayant deux valeurs distinctives possibles.

3.8.8 caractère: Élément d'un ensemble utilisé pour l'organisation, la commande ou la représentation des données.

NOTE – Ceci implique par exemple que le caractère de combinaison «accent aigu» et la minuscule «e» constituent deux caractères de la grille française ISO 646, et non pas un caractère unique «é».

3.8.9 syntaxe abstraite caractères: Toute syntaxe abstraite dont les valeurs sont toutes les chaînes composées de zéro, un ou plusieurs caractères appartenant à une collection de caractères donnée.

3.8.10 répertoire de caractères: Caractères d'un jeu de caractères indépendamment de toute considération quant à la manière dont ces caractères sont codés.

3.8.11 types chaîne de caractères: Types simples dont les valeurs sont des chaînes de caractères pris dans un jeu donné.

3.8.12 syntaxe de transfert de caractères: Toute syntaxe de transfert pour une syntaxe abstraite caractères.

NOTE – L'ASN.1 ne prend pas en charge les syntaxes de transfert de caractères qui ne codent pas toute chaîne de caractères sur un nombre entier d'octets.

3.8.13 type choix: Type défini par l'indication d'une liste de types distincts; chaque valeur du type choix dérive d'une valeur de l'un quelconque des types composants.

3.8.14 type composant: Un des types indiqués en référence dans une déclaration de type CHOICE (choix), SET (ensemble), SEQUENCE (séquence), SET OF (ensemble-de), ou SEQUENCE OF (séquence-de).

3.8.15 contrainte: Notation qui, associée à un type, permet d'en définir un sous-type.

3.8.16 caractères de contrôle: Caractères apparaissant dans certains répertoires de caractères et ayant reçu un nom (et éventuellement une fonction définie en relation avec certains environnements), mais qui ne se sont pas vus affecter un symbole graphique et qui ne sont pas non plus des caractères d'espacement.

NOTE – NEWLINE (nouvelle ligne) et TAB (tabulation) sont des exemples de caractères de contrôle qui se sont vus affecter des fonctions de formatage dans un environnement d'édition. DLE (échappement de transmission) est un exemple de caractère de contrôle qui s'est vu affecter une fonction dans un environnement de communication.

3.8.17 temps universel coordonné (UTC) (*coordinated universal time*): Echelle de temps conservée par le Bureau international de l'heure, et servant de base à la diffusion coordonnée des fréquences standards et des signaux horaires.

NOTES

1 L'origine de cette définition est la Recommandation 460-2 du Comité consultatif international des radiocommunications (CCIR). Le CCIR a également défini le sigle UTC du temps universel coordonné.

2 L'UTC et le temps moyen de Greenwich (GMT) (*Greenwich mean time*) sont deux normes de temps qui indiquent le même temps pour la plupart des applications pratiques.

3.8.18 élément: Membre d'une classe d'éléments, distinguable de tous les autres éléments de cette classe.

- 3.8.19 classe d'éléments:** Type (dont les éléments sont ses valeurs) ou classe d'objets informationnels (dont les éléments sont tous les objets possibles de cette classe).
- 3.8.20 ensemble d'éléments:** Un ou plusieurs éléments d'une même classe.
- 3.8.21 type pdv encapsulé (valeurs de données de présentation encapsulées):** Type dont l'ensemble des valeurs est la réunion des ensembles de valeurs dans toutes les syntaxes abstraites possibles. Ce type fait partie d'une spécification ASN.1 qui véhicule une valeur dont le type peut être défini extérieurement à cette spécification ASN.1. Il comporte également un identificateur du type de la valeur véhiculée ainsi qu'un identificateur de la règle de codage utilisée pour coder la valeur.
- 3.8.22 codage:** Séquence binaire résultant de l'application d'un ensemble de règles de codage à une valeur d'une syntaxe abstraite donnée.
- 3.8.23 règles de codage ASN.1:** Règles qui spécifient la représentation des valeurs de types ASN.1 durant leur transfert; elles permettent aussi de retrouver les valeurs à partir de leur représentation, une fois leur type connu.
- NOTE – Aux fins de la spécification des règles de codage, les différentes notations de types (et valeurs) données en référence, qui peuvent fournir d'autres notations pour des types (et valeurs) prédéfinis, ne sont pas applicables.
- 3.8.24 type énuméré:** Type simple dont chaque valeur reçoit un identificateur distinct dans le cadre de la notation du type.
- 3.8.25 type externe:** Type apparaissant dans une spécification ASN.1 et comportant une valeur dont le type peut être défini extérieurement à cette spécification. Le type externe comporte une identification du type de la valeur concernée.
- 3.8.26 référence externe:** Référence de type, référence de valeur, objet informationnel, etc. défini dans un module quelconque autre que celui dans lequel il y est fait référence, la référence à la définition s'effectuant en préfixant le nom du module de définition au nom de l'élément cité.
- EXEMPLE – NomModule.RéférenceType
- 3.8.27 Faux:** Une des deux valeurs distinctives du type booléen (voir «Vrai»).
- 3.8.28 gouvernant:** Type ou classe d'objets informationnels qui commande l'interprétation d'un objet, d'un ensemble d'objets, d'une valeur, d'un ensemble de valeurs ou d'un sous-type, en imposant aux éléments intervenant dans leur notation d'être des notations de valeurs respectivement de ce type ou de cette classe.
- 3.8.29 type entier:** Type simple dont les valeurs distinctives sont les entiers relatifs (les positifs, les négatifs, et l'élément nul en tant que valeur unique).
- NOTE – Les règles de codage particulières limitent l'intervalle de variation possible des entiers, mais ces limites sont choisies de façon à ne gêner en rien les utilisateurs de l'ASN.1.
- 3.8.30 items:** Séquences nommées de caractères du jeu de caractères ASN.1, spécifiées à l'article 9, et utilisées pour former la notation ASN.1.
- 3.8.31 module:** Une ou plusieurs instances d'utilisation de la notation ASN.1 pour la définition de types, de valeurs, etc., qui sont encapsulées au moyen de la notation de module ASN.1 (voir l'article 10).
- 3.8.32 type néant (Null):** Type simple comprenant une seule valeur, appelée «néant».
- 3.8.33 objet:** Élément bien défini d'information, de définition ou de spécification, nécessitant un nom afin d'en identifier l'intervention dans une instance de communication.
- 3.8.34 type descripteur d'objet:** Type dont les valeurs distinctives sont des textes en langage naturel décrivant brièvement un objet.
- NOTE – Une valeur de descripteur d'objet est généralement associée à un seul objet. Seule la valeur d'identificateur d'objet identifie sans ambiguïté l'objet.
- 3.8.35 identificateur d'objet:** Valeur (distincte de toutes les autres), associée à un objet.
- 3.8.36 type identificateur d'objet:** Type simple dont les valeurs distinctives sont l'ensemble de tous les identificateurs d'objet affectés conformément aux règles de la présente Recommandation | Norme internationale.
- NOTE – Les règles de la présente Recommandation | Norme internationale permettent à des autorités très diverses d'associer indépendamment les unes des autres des identificateurs à des objets.
- 3.8.37 type chaîne d'octets:** Type simple dont chaque valeur distinctive est une séquence ordonnée de zéro, un ou plusieurs octets (l'octet étant une séquence ordonnée de 8 bits).

3.8.38 notation de type ouvert: Notation ASN.1 servant à désigner un ensemble de valeurs appartenant à plus d'un type ASN.1.

NOTES

1 Les expressions «type ouvert» et «notation de type ouvert» sont synonymes dans le corps de la présente Recommandation | Norme internationale.

2 Les règles de codage de l'ASN.1 assurent toutes le codage non ambigu des valeurs appartenant à un type ASN.1 unique, mais elles n'assurent pas nécessairement le codage non ambigu d'une «notation de type ouvert», qui véhicule des valeurs de types ASN.1 qui ne sont pas encore normalement déterminés au moment de la spécification. Le type de valeur codée dans la «notation de type ouvert» doit être connu avant de pouvoir déterminer de manière non ambiguë la valeur abstraite de ce champ.

3 Dans la présente Recommandation | Norme internationale, la seule notation correspondant à un type ouvert est le type "ObjectClassFieldType" (type de champ de classe d'objets), spécifié dans la Rec. UIT-T X.681 | ISO/CEI 8824-2, et dans laquelle le nom de champ "FieldName" désigne soit un champ de type, soit un champ de valeur de type variable. La notation ANY ("quelconque"), définie dans la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990 et décrite dans l'Annexe I, était une notation de type ouvert.

3.8.39 type parent (d'un sous-type): Type dont dérive un sous-type par imposition de contraintes.

NOTE – Le type parent peut lui-même être un sous-type d'un autre type.

3.8.40 production: Partie de la notation formelle utilisée pour spécifier la notation ASN.1.

3.8.41 type réel: Type simple dont les valeurs distinctives (spécifiées à l'article 18) appartiennent à l'ensemble des réels.

3.8.42 définitions récursives: Ensemble de définitions en notation ASN.1 qui ne peuvent pas être réordonnées de telle sorte que tous les types utilisés dans une structure soient définis avant la définition de cette structure.

NOTE – Les définitions récursives sont autorisées en notation ASN.1: il appartient à l'utilisateur de s'assurer que les valeurs des types résultants ont une représentation finie.

3.8.43 type chaîne de caractères restreinte: Type de chaîne de caractères dont les caractères sont choisis dans un répertoire de caractères donné identifié dans la spécification du type.

3.8.44 type sélection: Type défini par référence à un type composant d'un type «choix», et dont les valeurs sont précisément celles de ce type composant.

3.8.45 type séquence: Type défini en désignant une liste ordonnée de types (dont certains peuvent être déclarés optionnels); chaque valeur du type séquence ainsi défini est une liste ordonnée de valeurs, une par type composant.

NOTE – Une valeur du type séquence ne doit pas nécessairement contenir une valeur d'un type composant si celui-ci est déclaré optionnel.

3.8.46 type séquence-de: Type défini en désignant un seul type composant; chaque valeur du type séquence-de ainsi défini est une liste ordonnée comportant zéro, une ou plusieurs valeurs du type composant.

3.8.47 type ensemble: Type défini en désignant une liste fixe, non ordonnée, de types distincts (dont certains peuvent être déclarés optionnels); chaque valeur du type ensemble est une liste non ordonnée de valeurs, une par type composant.

NOTE – Une valeur du type ensemble ne contiendra pas nécessairement la valeur d'un type composant si celui-ci est déclaré optionnel.

3.8.48 type ensemble-de: Type défini en désignant un seul type composant; chaque valeur du type ensemble-de est une liste non ordonnée comportant zéro, une ou plusieurs valeurs du type composant.

3.8.49 type simple: Type défini en extension (en spécifiant directement l'ensemble de ses valeurs).

3.8.50 caractère espacement: Caractère d'un répertoire destiné à être inclus en impression avec une chaîne de caractères graphiques, mais qui est représenté matériellement par un vide; il n'est généralement pas considéré comme un caractère de contrôle.

NOTE – Un répertoire de caractères peut comporter un caractère d'espacement, ou plusieurs de différentes classes.

3.8.51 sous-type (d'un type parent): Type dont les valeurs sont un sous-ensemble (ou l'ensemble complet) des valeurs d'un autre type (le type parent).

3.8.52 étiquette: Dénomination de type associée à chaque type ASN.1.

3.8.53 type étiqueté: Type défini par la désignation d'un type existant et d'une étiquette; le type étiqueté ainsi formé et le type existant sont isomorphes mais distincts.

3.8.54 étiquetage: Remplacement de l'étiquette existante (éventuellement l'étiquette par défaut) d'un type par une étiquette spécifiée.

3.8.55 Vrai: Une des deux valeurs distinctives du type booléen (voir «Faux»).

3.8.56 type: Ensemble nommé de valeurs.

3.8.57 nom de référence d'un type: Nom associé de manière unique à un type dans un contexte donné.

NOTE – Des noms de référence sont affectés aux types définis dans la présente Recommandation | Norme internationale; ils sont disponibles universellement en notation ASN.1. D'autres noms de référence sont définis dans diverses Recommandations | Normes internationales et ne sont alors applicables que dans le contexte de celles-ci.

3.8.58 type chaîne de caractères non restreinte: Type dont les valeurs sont celles d'une syntaxe abstraite de caractères identifiée séparément pour chaque instance d'utilisation de ce type.

3.8.59 utilisateur (de la notation ASN.1): Personne physique ou morale qui définit la syntaxe abstraite d'un élément d'information particulier en notation ASN.1.

3.8.60 valeur: Élément distinctif d'un ensemble de valeurs.

3.8.61 nom de référence de valeur: Nom associé de manière unique à une valeur dans un contexte donné.

3.8.62 ensemble de valeurs: Collection de valeurs d'un type donné; cet ensemble est sémantiquement équivalent à un sous-type.

3.8.63 blanc: Toute action de formatage se traduisant par un espace blanc sur la page d'impression, par exemple un espacement, une tabulation, ou l'utilisation répétée de tels caractères.

4 Abréviations

ASN.1	Notation de syntaxe abstraite numéro un (<i>abstract syntax notation one</i>)
BER	Règles de codage de base de l'ASN.1 (<i>basic encoding rules</i>)
DCC	Indicatif de pays pour la transmission de données (<i>data country code</i>)
DNIC	Code d'identification de réseau pour données (<i>data network identification code</i>)
ICD	Prescripteur de code international (<i>international code designer</i>)
CEI	Commission électrotechnique internationale
ISO	Organisation internationale de normalisation (<i>international standards organization</i>)
UIT-T	Union internationale des télécommunications – Secteur de la normalisation des télécommunications
PDV	Valeur de données de présentation (<i>presentation data value</i>)
PER	Règles de codage compact de l'ASN.1 (<i>packed encoding rules of ASN.1</i>)
ER	Exploitation reconnue
UCS	Jeu de caractères universels codés sur multi-octets (<i>universal multiple-octet coded character set</i>)

5 Notation

Une notation ASN.1 consiste en une séquence de caractères prise dans le jeu de caractères ASN.1 spécifié à l'article 8.

Chaque instance de notation ASN.1 contient des caractères du jeu ASN.1 regroupés en items. L'article 9 spécifie toutes les séquences de caractères formant des items ASN.1, avec le nom de ces items.

La notation ASN.1 est définie à l'article 10 (et les articles suivants), en spécifiant la collection des séquences d'items qui forment des instances valides de la notation ASN.1, avec la sémantique de chaque séquence.

Pour spécifier ces ensembles, la présente Recommandation | Norme internationale utilise une notation formelle définie dans les points suivants.

5.1 Productions

Une collection nouvelle (plus complexe) de séquences ASN.1 est définie au moyen d'une production. Une production utilise les noms des collections de séquences de production définies dans la présente Recommandation | Norme internationale, et forme une nouvelle collection de séquences de production en spécifiant:

- a) soit que la nouvelle collection de séquences sera composée de n'importe quelle séquence contenue dans l'une quelconque des collections d'origine;
- b) soit que la nouvelle collection sera composée d'une séquence qui peut être générée en prenant une séquence et une seule dans chaque collection puis en les juxtaposant dans un ordre spécifié.

Chaque production comporte les parties suivantes, dans l'ordre, sur une ou plusieurs lignes:

- a) le nom de la nouvelle collection de séquences de production;
- b) les caractères
 ::=
- c) une ou plusieurs collections au choix de séquences de production, telles qu'elles sont définies au 5.2, séparées par le caractère
 |

Une séquence de production figure dans la nouvelle collection si elle figure dans une ou plusieurs des collections au choix. La nouvelle collection est désignée dans la présente Recommandation | Norme internationale par le nom mentionné à l'alinéa a) ci-dessus.

NOTE – Si la même séquence figure dans plusieurs collections de séquences au choix du membre droit de la production, toute ambiguïté sémantique de la notation résultante est résolue par d'autres parties de la séquence ASN.1 complète.

5.2 Collections au choix d'une production

Chacune des collections de séquences de production de la structure «une ou plusieurs collections au choix de» (voir 5.1 c)) est spécifiée par une liste de noms. Chaque nom est soit le nom d'un item, soit le nom d'une collection de séquences de production définie par une production dans la présente Recommandation | Norme internationale.

La collection de séquences de production définie dans la structure à plusieurs formes au choix comprend toutes les séquences de production obtenues en prenant l'une quelconque des séquences de production (ou des items) associées au premier nom, combinée avec (et suivie de) l'une quelconque des séquences de production (ou items) associées au second nom, combinée avec (et suivie de) l'une quelconque des séquences de production (ou items) associées au troisième nom, et ainsi de suite jusques et y compris le dernier nom (ou item) de la structure à plusieurs choix.

5.3 Exemple de production

```
BitStringValue ::=
    bstring |
    hstring |
    "{" IdentifierList "}"
```

est une production qui associe au nom "BitStringValue" une des séquences de production suivantes:

- a) soit une chaîne binaire quelconque "bstring" (un item);
- b) soit une chaîne hexadécimale quelconque "hstring" (un item);
- c) soit une séquence de production quelconque associée à la liste d'identificateurs "IdentifierList", précédée d'une "{" et suivie d'une "}".

NOTE – "{" et "}" sont les noms des items contenant respectivement les seuls caractères { et } (voir 9.15).

Dans cet exemple, "IdentifierList" serait défini par une autre production, placée avant ou après la production définissant "BitStringValue".

5.4 Mise en page

Chaque production de la présente Recommandation | Norme internationale est précédée et suivie d'une ligne blanche. Il n'y a pas de ligne blanche à l'intérieur des productions. Les productions peuvent occuper une ou plusieurs lignes. La mise en page n'est pas significative.

5.5 Récursivité

Les productions de la présente Recommandation | Norme internationale sont souvent récursives. Dans ce cas, les productions doivent être réappliquées autant de fois qu'il est nécessaire, jusqu'à ce qu'aucune nouvelle séquence ne soit générée.

NOTE – Dans de nombreux cas, cette réapplication produit une collection non bornée de séquences autorisées, certaines pouvant elles-mêmes être non bornées. Ceci n'est pas une erreur.

5.6 Pointage d'une collection de séquences

La présente Recommandation | Norme internationale permet de pointer une collection de séquences (faisant partie de la notation ASN.1) en citant le nom du membre gauche d'une production (à gauche du signe ::=); le nom est mis entre guillemets (") pour le distinguer du texte en langage naturel, sauf s'il apparaît à l'intérieur d'une production.

5.7 Pointage d'un item

La présente Recommandation | Norme internationale permet de pointer un item en citant son nom; le nom est mis entre guillemets (") pour le distinguer du texte en langage naturel, sauf s'il apparaît à l'intérieur d'une production et qu'il ne s'agit ni d'un élément monocaractère ni des signes ":", ".", ou "...".

5.8 Notations abrégées

Pour rendre les productions plus concises et en faciliter la lecture, les notations abrégées suivantes sont utilisées dans les définitions des collections de séquences de production ASN.1 dans les Rec. UIT-T X.681 | ISO/CEI 8824-2, UIT-T X.682 | ISO/CEI 8824-3 et UIT-T X.683 | ISO/CEI 8824-4 (mais elles ne sont utilisées nulle part dans la présente Recommandation | Norme internationale):

- a) un astérisque (*) placé après deux noms "A" et "B" indique soit l'item vide (empty) (voir 9.7), soit une séquence de production associée à "A", soit une suite de séquences de productions ASN.1 associées alternativement à "A" et à "B" en commençant et en terminant par une séquence de production associée à "A". Ainsi:

C ::= A B *

équivalent à

C ::= D | empty

D ::= A | A B D

"D" étant une variable auxiliaire n'apparaissant pas ailleurs dans les productions.

EXEMPLE – "C ::= A B *" est une notation abrégée désignant l'une quelconque des productions suivantes:

empty (*vide*)

A

A B A

A B A B A

A B A B A B A

...

- b) un signe plus (+) est semblable à l'astérisque en a), sauf que l'item vide est exclus. Ainsi:

E ::= A B +

équivalent à

E ::= A | A B E

EXEMPLE – "E ::= A B +" est une notation abrégée désignant l'une quelconque des productions suivantes:

A

A B A

A B A B A

A B A B A B A

...

- c) un point d'interrogation (?) placé à la suite d'un nom indique soit l'item vide (voir 9.7), soit une séquence de production associée à ce nom. Ainsi:

F ::= A ?

équivalent à

F ::= empty | A

6 Etiquettes

6.1 On spécifie une étiquette en indiquant une classe et un numéro dans cette classe. L'étiquette peut appartenir à l'une des classes suivantes:

- universelle;
- application;
- privée;
- propre au contexte.

6.2 Le numéro est un entier non négatif, spécifié en notation décimale.

L'article 28 spécifie les restrictions applicables aux étiquettes par l'utilisateur ASN.1.

Le Tableau 1 récapitule les étiquettes de la classe universelle affectées dans la présente Recommandation | Norme internationale.

Tableau 1 – Affectation des étiquettes de la classe universelle

UNIVERSAL 0	Réservée à l'utilisation par les règles de codage
UNIVERSAL 1	Type booléen
UNIVERSAL 2	Type entier
UNIVERSAL 3	Type chaîne binaire
UNIVERSAL 4	Type chaîne d'octets
UNIVERSAL 5	Type néant
UNIVERSAL 6	Type identificateur d'objet
UNIVERSAL 7	Type descripteur d'objet
UNIVERSAL 8	Type externe et type instance-de
UNIVERSAL 9	Type réel
UNIVERSAL 10	Type énuméré
UNIVERSAL 11	Type pdv (valeur de donnée de présentation) encapsulé
UNIVERSAL 12-15	Réservées pour de futures éditions de la présente Recommandation Norme internationale
UNIVERSAL 16	Types séquence et séquence-de
UNIVERSAL 17	Types ensemble et ensemble-de
UNIVERSAL 18-22, 25-30	Types chaînes de caractères
UNIVERSAL 23-24	Types temps
UNIVERSAL 31 à ...	Réservées aux addenda à la présente Recommandation Norme internationale

6.3 Certaines règles de codage nécessitent de disposer d'un ordre canonique pour les étiquettes. Pour en assurer l'uniformité, un tel ordre canonique a été défini au 6.4.

NOTE – Cet ordre n'est pas utilisé dans la présente Recommandation | Norme internationale, mais d'autres Recommandations | Normes internationales s'y réfèrent.

6.4 L'ordre canonique des étiquettes est défini de la manière suivante:

- a) les éléments ou notations portant des étiquettes de la classe universelle apparaîtront en premier, suivis de ceux portant des étiquettes de la classe application, suivis de ceux portant des étiquettes propres au contexte, suivis de ceux portant des étiquettes de la classe privée;
- b) à l'intérieur de chaque classe d'étiquettes, les éléments ou notations apparaîtront dans l'ordre croissant de leur numéro d'étiquette.

7 Utilisation de la notation ASN.1

7.1 Un type est défini en ASN.1 par la notation "Type" (voir 14.1).

7.2 Une valeur d'un type donné est déclarée en ASN.1 par la notation "Value" (voir 14.7).

NOTE – Il n'est en général pas possible d'interpréter la notation d'une valeur sans en connaître le type.

7.3 Un type est affecté à un nom de référence de type en ASN.1 par la notation "TypeAssignment" (voir 13.1), "ValueSetTypeAssignment" (voir 13.4), "ParameterizedTypeAssignment" (voir 8.2 de la Rec. UIT-T X.683 | ISO/CEI 8824-4) ou "ParameterizedValueSetTypeAssignment" (voir 8.2 de la Rec. UIT-T X.683 | ISO/CEI 8824-4).

7.4 Une valeur est affectée à un nom de référence de valeur en ASN.1 par la notation "ValueAssignment" (voir 13.2) ou "ParameterizedValueAssignment" (voir 8.2 de la Rec. UIT-T X.683 | ISO/CEI 8824-4).

7.5 Les différentes productions de notation d'affectation "Assignment" ne seront utilisées que dans la notation "ModuleDefinition" (définition de module) (excepté ce qui est spécifié en Note 2 du 10.1).

8 Jeu de caractères ASN.1

8.1 Un item ASN.1 consiste en une séquence de caractères pris dans le Tableau 2, sous réserve des indications spécifiées aux 8.2 et 8.3.

Tableau 2 – Caractères ASN.1

A à Z
a à z
0 à 9
:= , { } < . @ () [] - ' & ^ * ; !

NOTE – Quand des normes dérivées équivalentes sont élaborées par des organismes de normalisation nationaux, des caractères supplémentaires peuvent apparaître dans les items suivants:

typereference (référence de type) (voir 9.2)

identifieur (identificateur) (voir 9.3)

valuereference (référence de valeur) (voir 9.4)

modulereference (référence de module) (voir 9.5)

Quand des caractères supplémentaires sont introduits pour pouvoir noter un langage dans lequel il n'y a pas de distinction entre majuscules et minuscules, la distinction syntaxique assurée en imposant au premier caractère de certains items ASN.1 d'être une majuscule ou une minuscule sera obtenue d'une autre façon. Cette disposition est introduite pour permettre d'écrire des spécifications ASN.1 valides dans différentes langues.

8.2 Lorsque la notation est utilisée pour spécifier la valeur d'un type chaîne de caractères, tous les symboles graphiques du jeu défini peuvent apparaître entre guillemets (") dans la notation ASN.1 (voir 9.11).

8.3 Des symboles graphiques supplémentaires quelconques peuvent apparaître dans l'item commentaire (voir 9.6).

8.4 Aucune signification particulière ne sera accordée au style, à la taille, à la couleur, à la graisse ou aux autres caractéristiques typographiques d'affichage.

8.5 Les majuscules et les minuscules seront considérées comme distinctes.

9 Items ASN.1

9.1 Règles générales

9.1.1 Les paragraphes suivants spécifient les caractères utilisés pour les items ASN.1. Dans chaque cas, le nom de l'item est donné avec la définition des séquences de caractères qui le forment.

9.1.2 Chaque item spécifié dans les points suivants (exception faite des items "bstring", "hstring" et "cstring") apparaîtra sur une seule ligne et ne contiendra pas de blancs (exception faite des items "comment", "bstring", "hstring" et "cstring") (voir 9.9, 9.10 et 9.11).

9.1.3 La longueur d'une ligne n'est pas limitée.

9.1.4 Les items des séquences de production spécifiées par la présente Recommandation | Norme internationale (la notation ASN.1) pourront apparaître sur une ou plusieurs lignes et être séparés par un blanc, des interlignes et des commentaires.

9.1.5 Si le ou les caractères initiaux d'un item font partie des caractères qu'il est permis d'ajouter à la suite de la chaîne de caractères de l'item précédent, ces deux items devront être séparés par un blanc, un interligne ou un commentaire.

9.2 Référence de type

Nom de l'item – typereference

9.2.1 Une référence "typereference" est constituée d'un nombre quelconque (un ou plusieurs) de lettres, chiffres et traits d'union. Le premier caractère doit être une majuscule. Le dernier caractère ne peut pas être un trait d'union. Un trait d'union ne doit pas être immédiatement suivi par un autre trait d'union.

NOTE – Les règles concernant le trait d'union sont destinées à éviter toute confusion possible avec un commentaire (éventuellement placé à la suite).

9.2.2 Une référence "typereference" ne doit pas être l'une des séquences de caractères réservées énumérées au 9.16.

9.3 Identificateur

Nom de l'item – identifier

Un identificateur "identifier" sera constitué d'un nombre quelconque (un ou plusieurs) de lettres, chiffres et traits d'union. Le premier caractère doit être une minuscule. Le dernier caractère ne peut pas être un trait d'union. Un trait d'union ne peut pas être immédiatement suivi par un autre trait d'union.

NOTE – Les règles concernant le trait d'union sont destinées à éviter toute confusion possible avec un commentaire (éventuellement placé à la suite).

9.4 Référence de valeur

Nom de l'item – valuereference

Une référence "valuereference" sera constituée de la séquence de caractères spécifiée pour un identificateur au 9.3. Lors de l'analyse d'une instance de cette notation, une référence "valuereference" sera distinguée d'un identificateur "identifier" par le contexte dans lequel elle apparaît.

9.5 Référence de module

Nom de l'item – modulereference

Une référence "modulereference" sera constituée de la séquence de caractères spécifiée pour une référence "typereference" au 9.2. Lors de l'analyse d'une instance de cette notation, une référence "modulereference" sera distinguée d'une référence "typereference" par le contexte dans lequel elle apparaît.

9.6 Commentaire

Nom de l'item – comment

9.6.1 Un commentaire "comment" n'est jamais pointé dans la définition d'une notation ASN.1: il peut toutefois apparaître n'importe où entre d'autres items ASN.1, et n'a pas de signification syntaxique.

NOTE – Dans le contexte d'une Recommandation | Norme internationale comportant des déclarations ASN.1, un commentaire ASN.1 peut néanmoins contenir un texte ayant force de norme se rapportant à la sémantique de l'application ou à des contraintes syntaxiques.

9.6.2 Un commentaire commence par un double trait d'union et se termine au premier double trait d'union rencontré ou en fin de la ligne. Un commentaire ne doit pas contenir de doubles traits d'union autres que ceux par lesquels il commence et éventuellement se termine. Il peut inclure des symboles graphiques qui n'appartiennent pas au jeu de caractères spécifié au 8.1 (voir 8.3).

9.7 Item vide

Nom de l'item – empty

L'item vide "empty" ne contient aucun caractère. Il est utilisé dans la notation de l'article 5 quand plusieurs ensembles de séquences au choix sont spécifiés, pour inclure l'ensemble vide parmi les séquences possibles.

9.8 Item numéro

Nom de l'item – number

Un numéro "number" comprend un ou plusieurs chiffres. Le premier chiffre doit être différent de zéro, sauf si le numéro n'a qu'un seul chiffre.

NOTE – L'item "number" est toujours interprété comme un entier en base dix.

9.9 Item chaîne binaire

Nom de l'item – bstring

Une chaîne binaire "bstring" se compose d'un nombre quelconque (éventuellement zéro) de 0, de 1, de blancs et d'interlignes, précédés du caractère (') et suivis des deux caractères

'B

Les blancs et les interlignes à l'intérieur d'un item chaîne binaire n'ont pas de signification.

EXEMPLE – '01101100'B

9.10 Item chaîne hexadécimale

Nom de l'item – hstring

9.10.1 Une chaîne "hstring" est composée d'un nombre quelconque (éventuellement zéro) de blancs, d'interlignes et de caractères pris parmi:

0 1 2 3 4 5 6 7 8 9 A B C D E F

précédés du caractère (') et suivis des deux caractères

'H

Les blancs et les interlignes à l'intérieur d'un item chaîne hexadécimale n'ont pas de signification.

EXEMPLE – 'AB0196'H

9.10.2 Chaque caractère représente la valeur d'un demi-octet exprimée en notation hexadécimale.

9.11 Item chaîne de caractères

Nom de l'item – cstring

9.11.1 Une chaîne de caractères "cstring" est composée d'un nombre quelconque (éventuellement zéro) de symboles graphiques et de caractères d'espacement du jeu désigné par le type de la chaîne de caractères, placés entre guillemets (""). Si le jeu de caractères comprend le guillemet, il sera représenté le cas échéant dans la chaîne de caractères par deux guillemets consécutifs ("") sur la même ligne et sans espace intermédiaire. La chaîne de caractères peut occuper plusieurs lignes de texte, auquel cas la valeur de chaîne de caractères représentée ne comportera pas de caractère d'espacement dans la position située juste avant ou après les fins de ligne de sa représentation. Un blanc situé immédiatement avant ou après une fin de ligne d'une chaîne "cstring" n'a pas de signification.

NOTES

1 Une chaîne "cstring" ne peut être utilisée que pour représenter des chaînes dont chaque caractère est soit un symbole graphique soit un caractère d'espacement. Il existe d'autres syntaxes ASN.1 pour représenter des chaînes comportant des caractères de contrôle (voir article 32).

2 Une chaîne "cstring" est constituée des caractères associées aux symboles graphiques et aux caractères d'espacement imprimables. Les caractères d'espacement placés immédiatement avant ou après une fin de ligne de la chaîne "cstring" ne font pas partie de la chaîne représentée (ils sont ignorés). Lorsque la chaîne "cstring" comporte des caractères d'espacement ou lorsque les symboles graphiques du répertoire de caractères comportent des ambiguïtés, la chaîne de caractères représentée par une chaîne "cstring" pourra être ambiguë.

EXEMPLE 1 – "屎屍市弑"

EXEMPLE 2 – La chaîne "cstring"

```
"ABCDE FGH  
IJK""XYZ"
```

peut être utilisée pour représenter une valeur de chaîne de caractères du type IA5String. La valeur représentée est constituée des caractères

```
ABCDE FGHIJK"XYZ"
```

où l'espacement voulu entre E et F peut être ambigu si une police de caractères à chasse variable est utilisée dans la spécification (ce qui est le cas dans l'exemple).

9.11.2 Un caractère de combinaison sera représenté dans une chaîne "cstring" comme un caractère isolé et ne recevra pas en surimpression le caractère avec lequel il se combine. (Ceci garantit la détermination sans ambiguïté de l'ordre des caractères de combinaison dans la valeur de chaîne.)

EXEMPLE – L'accent aigu de combinaison et la minuscule "e" sont deux caractères de la grille ISO 646 "Version française", et seront donc représentés dans la chaîne "cstring" correspondante par deux caractères et non par le simple caractère "é".

9.11.3 La chaîne "cstring" ne sera pas utilisée pour représenter des chaînes de caractères contenant des caractères de contrôle. Seuls les caractères graphiques et les caractères d'espacement sont permis.

9.12 Item affectation

Nom de l'item – "::~="

Cet item est composé de la séquence de caractères

```
::=
```

NOTE – Cette séquence ne doit contenir aucun caractère blanc (voir 9.1.2).

9.13 Séparateur de plage

Nom de l'item – ".."

Cet item se compose de la séquence de caractères

```
..
```

NOTE – Cette séquence ne doit contenir aucun caractère blanc (voir 9.1.2).

9.14 Points de suspension

Nom de l'item – "..."

Cet item se compose de la séquence de caractères

...

NOTE – Cette séquence ne doit contenir aucun caractère blanc (voir 9.1.2).

9.15 Items à caractère unique

Noms des items:

"{"
 "}"
 "<"
 ","
 ";"
 "("
 ")"
 "["
 "]"
 "-" (trait d'union)
 ":"
 ">"
 "@"
 "|"
 "!"
 "^"

Un item dont le nom est l'un de ceux indiqués ci-dessus est composé du seul caractère sans les guillemets.

9.16 Items mots réservés

Noms des mots réservés:

ABSENT	EMBEDDED	INSTANCE	REAL
ABSTRACT-SYNTAX	END	INTEGER	SEQUENCE
ALL	ENUMERATED	INTERSECTION	SET
APPLICATION	EXCEPT	ISO646String	SIZE
AUTOMATIC	EXPLICIT	MAX	STRING
BEGIN	EXPORTS	MIN	SYNTAX
BIT	EXTERNAL	MINUS-INFINITY	T61String
BMPString	FALSE	NULL	TAGS
BOOLEAN	FROM	NumericString	TeletexString
BY	GeneralizedTime	OBJECT	TRUE
CHARACTER	GeneralString	ObjectDescriptor	UNION
CHOICE	GraphicString	OCTET	UNIQUE
CLASS	IA5String	OF	UNIVERSAL
COMPONENT	TYPE-IDENTIFIER	OPTIONAL	UniversalString
COMPONENTS	IDENTIFIER	PDV	UTCTime
CONSTRAINED	IMPLICIT	PLUS-INFINITY	VideotexString
DEFAULT	IMPORTS	PRESENT	VisibleString
DEFINITIONS	INCLUDES	PrintableString	WITH
		PRIVATE	

Les items dont les noms figurent ci-dessus sont composés de la séquence des caractères du nom et sont des séquences réservées.

NOTES

1 Aucun blanc ne doit figurer dans ces séquences.

2 Les mots clés CLASS, CONSTRAINED, INSTANCE, SYNTAX et UNIQUE ne sont pas utilisés dans la présente Recommandation | Norme internationale; ils le sont dans les Rec.UIT-T X.681 | ISO/CEI 8824-2, UIT-T X.682 | ISO/CEI 8824-3 et UIT-T X.683 | ISO/CEI 8824-4.

10 Définition de module

10.1 Une définition "ModuleDefinition" est spécifiée par les productions suivantes:

```

ModuleDefinition ::=
    ModuleIdentifieur
    DEFINITIONS
    TagDefault
    "::="
    BEGIN
    ModuleBody
    END

ModuleIdentifieur ::=
    modulereference
    DefinitiveIdentifieur

DefinitiveIdentifieur ::=
    "{" DefinitiveObjIdComponentList "}" | empty

DefinitiveObjIdComponentList ::=
    DefinitiveObjIdComponent |
    DefinitiveObjIdComponent DefinitiveObjIdComponentList

DefinitiveObjIdComponent ::=
    NameForm |
    DefinitiveNumberForm |
    DefinitiveNameAndNumberForm

DefinitiveNumberForm ::= number

DefinitiveNameAndNumberForm ::= identifieur "(" DefinitiveNumberForm ")"

TagDefault ::=
    EXPLICIT TAGS |
    IMPLICIT TAGS |
    AUTOMATIC TAGS |
    empty

ModuleBody ::=
    Exports Imports AssignmentList |
    empty

Exports ::=
    EXPORTS SymbolsExported ";" |
    empty

SymbolsExported ::=
    SymbolList |
    empty

Imports ::=
    IMPORTS SymbolsImported ";" |
    empty

SymbolsImported ::=
    SymbolsFromModuleList |
    empty

SymbolsFromModuleList ::=
    SymbolsFromModule |
    SymbolsFromModuleList SymbolsFromModule

SymbolsFromModule ::=
    SymbolList FROM GlobalModuleReference

GlobalModuleReference ::=
    modulereference AssignedIdentifieur

AssignedIdentifieur ::=
    ObjectIdentifieurValue |
    DefinedValue |
    empty

```

```

SymbolList ::=
    Symbol |
    SymbolList "," Symbol

Symbol ::=
    Reference |
    ParameterizedReference

Reference ::=
    typereference |
    valuereference |
    objectclassreference |
    objectreference |
    objectsetreference

AssignmentList ::=
    Assignment |
    AssignmentList Assignment

Assignment ::=
    TypeAssignment |
    ValueAssignment |
    ValueSetTypeAssignment |
    ObjectClassAssignment |
    ObjectAssignment |
    ObjectSetAssignment |
    ParameterizedAssignment

```

NOTES

1 L'utilisation d'une référence paramétrée "ParameterizedReference" dans les listes EXPORTS et IMPORTS est spécifiée dans la Rec. UIT-T X.683 | ISO/CEI 8824-4.

2 Lorsque des notations sont données à titre d'exemple (et lors de la définition de types d'étiquettes de classe universelle dans la présente Recommandation | Norme internationale) le corps de module "ModuleBody" peut être utilisé à l'extérieur d'une définition de module "ModuleDefinition".

3 Les productions "TypeAssignment" et "ValueAssignment" sont spécifiées à l'article 13.

4 Le regroupement de types de données ASN.1 en modules ne détermine pas nécessairement le regroupement des valeurs de données de présentation dans des syntaxes abstraites nommées pour la définition de contextes de présentation.

5 La valeur de l'étiquette par défaut "TagDefault" de la définition d'un module affecte uniquement les types définis explicitement dans ce module. Elle n'affecte pas l'interprétation des types importés.

6 Le point virgule ";" n'apparaît ni dans la spécification de la liste d'affectation "AssignmentList" ni dans l'une quelconque de ses productions subordonnées; son utilisation est réservée aux développeurs d'outils ASN.1.

10.2 Si l'étiquette par défaut "TagDefault" est vide, elle est prise par défaut comme étant "EXPLICIT TAGS".

NOTE – L'article 28 donne la signification des étiquetages explicites, implicites et automatiques: "EXPLICIT TAGS", "IMPLICIT TAGS" et "AUTOMATIC TAGS".

10.3 Si l'étiquette par défaut "TagDefault" a la valeur "AUTOMATIC TAGS", on dit que l'étiquetage automatique a été choisi pour le module; sinon, on dit qu'il n'a pas été choisi. L'étiquetage automatique est une transformation syntaxique qui est appliquée (avec des conditions additionnelles) aux productions "ComponentTypeList" et "AlternativeTypeList" apparaissant dans la définition du module. Cette transformation est spécifiée formellement par 22.5, 24.3 et 26.2 pour ce qui est respectivement des types séquences, des types ensembles et des types choix.

10.4 La référence "modulereference" apparaissant dans la production "ModuleIdentifieur" est appelée nom du module.

NOTE – La possibilité de définir un même module ASN.1 comportant plusieurs corps de module "ModuleBody" partageant une même référence de module "modulereference" était autorisée (quoique déconseillée) dans les spécifications précédentes. Cette notation est interdite par la présente Recommandation | Norme internationale.

10.5 Les noms de module ne doivent être utilisés qu'une seule fois (sauf dans les cas spécifiés au 10.8) dans le domaine de visibilité de la définition du module.

10.6 Si l'identificateur définitif "DefinitiveIdentifieur" n'est pas vide, la valeur de l'identificateur d'objet indiquée identifie de manière unique et non ambiguë le module ainsi défini. Aucune valeur définie ne peut être utilisée pour définir la valeur de l'identificateur d'objet.

NOTE – Le problème de savoir quel volume de modifications impose l'adoption d'un nouvel identificateur "DefinitiveIdentifieur" n'est pas traité dans la présente Recommandation | Norme internationale.

10.7 Si l'identificateur affecté "AssignedIdentifieur" n'est pas vide, le module à partir duquel les items sont importés sera identifié de manière unique et non ambiguë soit par la valeur d'identificateur d'objet "ObjectIdentifieurValue", soit par la valeur définie "DefinedValue". Si c'est la valeur "DefinedValue" qui assure l'identification, elle devra être du type identificateur d'objet. Chaque référence de valeur "valuereference" apparaissant textuellement dans un identificateur "AssignedIdentifieur" satisfera à l'une des règles suivantes:

- a) elle est définie dans la liste d'affectation "AssignmentList" du module ainsi défini, et toutes les références de valeur "valuereference" apparaissant textuellement dans le membre droit de l'affectation satisfont aussi à la présente règle (règle "a") ou à la suivante (règle "b");
- b) elle apparaît en tant que symbole "Symbol" dans un module "SymbolsFromModule", dont l'identificateur affecté "AssignedIdentifieur" ne contient textuellement pas de référence de valeur "valuereference".

NOTE – Il est recommandé d'affecter au module un identificateur d'objet de manière à ce que d'autres modules puissent s'y référer sans ambiguïté.

10.8 La référence globale de module "GlobalModuleReference" du module d'origine dans la notation des symboles importés "SymbolsFromModule" apparaîtra dans la définition "ModuleDefinition" d'un autre module, sauf que si elle comporte un identificateur "DefinitiveIdentifieur" non vide, la référence de module "modulereference" peut être différente dans ces deux cas.

NOTE – On ne doit utiliser une référence de module "modulereference" différente de celle utilisée dans l'autre module que lorsqu'il faut importer les symboles de deux modules portant le même nom (l'appellation des modules n'ayant pas respecté les dispositions du 10.5). L'utilisation d'autres noms distincts permet d'utiliser ces noms dans le corps du module (voir 10.14).

10.9 Lorsqu'on utilise à la fois une référence de module "modulereference" et un identificateur affecté "AssignedIdentifieur" non vide pour désigner un module, ce dernier sera considéré comme définitif.

10.10 Lorsque le module désigné possède un identificateur définitif "DefinitiveIdentifieur" non vide, la référence globale "GlobalModuleReference" pointant vers ce module possédera un identificateur affecté "AssignedIdentifieur" non vide.

10.11 Lorsque la notation d'exportation "Exports" se voit affecter des symboles exportés "SymbolsExported":

- a) chaque "Symbol" de la liste "SymbolsExported" satisfera à une et une seule des deux conditions suivantes:
 - i) soit être défini dans le module en cours de constitution,
 - ii) soit apparaître une et une seule fois dans la liste "SymbolsImported" de la déclaration "Imports".
- b) tous les symboles qu'il serait utile de pouvoir pointer depuis l'extérieur du module seront inclus dans la liste "SymbolsExported", et seront seuls à pouvoir être ainsi pointés;
- c) lorsqu'il n'existe pas de tels symboles, la liste "SymbolsExported" (et non pas la déclaration "Exports") se verra affecter la valeur vide.

10.12 Lorsque les exportations "Exports" se voient affecter la valeur vide, tous les symboles "Symbol" définis dans le module peuvent être pointés par d'autres modules.

NOTE – La valeur vide est incluse dans les possibilités d'affectation de la déclaration "Exports" pour des raisons de compatibilité amont.

10.13 Les identificateurs apparaissant dans une liste "NamedNumberList", "Enumeration" ou "NamedBitList" sont implicitement exportés si la référence de type "typereference" les définissant est exportée ou apparaît en tant que composante (ou sous-composante) d'un type exporté.

10.14 Lorsque la notation d'importation "Imports" se voit affecter des symboles importés "SymbolsImported", toutes les conditions suivantes devront être remplies:

- a) chacun des symboles apparaissant dans une déclaration "SymbolsFromModule" soit sera défini dans le corps du module désigné par la référence globale "GlobalModuleReference" de la déclaration "SymbolsFromModule", soit figurera dans une déclaration IMPORTS de ce module. L'importation d'un "Symbol" figurant dans une déclaration IMPORTS du module pointé n'est autorisée que s'il existe une seule occurrence de ce "Symbol" dans la déclaration d'importation du module pointé et que le "Symbol" n'est pas défini dans ce module;

NOTE 1 – Ceci n'interdit pas d'importer dans un module deux symboles de même nom définis dans deux modules différents. Mais si le même nom de "Symbol" figure plus d'une fois dans la déclaration IMPORTS d'un module A, ce symbole ne peut plus être exporté de A pour être importé par un autre module B.

- b) Si, dans la définition du module extérieur désigné par la référence "GlobalModuleReference" de la déclaration d'origine des symboles "SymbolsFromModule" du module importateur, la déclaration d'exportation "Exports" comporte une liste de symboles exportés "SymbolsExported", le symbole devra apparaître dans la liste "SymbolsExported" de ce module extérieur;

- c) seuls les symboles figurant dans une liste "SymbolList" d'une production "SymbolsFromModule" (symboles tirés de modules) peuvent apparaître dans une déclaration de référence externe "External<x>Reference" (<x> remplaçant ici "value", "type", "object", "objectclass" ou "objectset") pointant sur le même module que la référence "modulereference" de la déclaration "GlobalModuleReference" de la production "SymbolsFromModule";
- d) lorsqu'il n'existe pas de symbole à importer, la valeur vide sera affectée à la liste "SymbolsImported";
 - NOTE 2 – Une des conséquences de c) et d) est que la déclaration "IMPORTS;" implique que le module ne peut pas contenir une référence externe "External<X>Reference".
- e) les productions "SymbolsFromModule" de la liste "SymbolsFromModuleList" comporteront chacune une référence globale "GlobalModuleReference" telle que:
 - i) les valeurs de référence "modulereference" qui y sont indiquées soient toutes différentes les unes des autres, et différentes également de la valeur "modulereference" associé au module importateur;
 - ii) les identificateurs affectés "AssignedIdentifier" désignent, lorsqu'ils ne sont pas vides, des valeurs d'identificateurs d'objet toutes différentes les unes des autres, et différentes également de la valeur d'identificateur d'objet (s'il y en a une) associée au module importateur.

10.15 Lorsque les importations "Imports" se voient affecter la valeur vide, le module peut quand même faire référence à des symboles définis dans d'autres modules au moyen de déclarations "External<x>Reference".

NOTE – La valeur vide est incluse dans les possibilités d'affectation de la déclaration "Imports" pour des raisons de compatibilité amont.

10.16 Les identificateurs apparaissant dans une liste "NamedNumberList", "Enumeration" ou "NamedBitList" sont implicitement importés si la référence de type "typereference" les définissant est importée ou apparaît en tant que composante (ou sous-composante) d'un type exporté.

10.17 Un "Symbol" énuméré dans une déclaration "SymbolsFromModule" peut apparaître dans le corps de module "ModuleBody" en tant que référence "Reference". La signification associée au symbole est celle qu'il a dans le module désigné par la notation "GlobalModuleReference" correspondante.

10.18 Lorsque un symbole donné apparaît également dans une liste d'affectation "AssignmentList" (ce qui est déconseillé) ou apparaît dans une ou plusieurs autres instances de déclaration "SymbolsFromModule", il ne sera utilisé que dans une référence "External<x>Reference". Autrement, il peut être utilisé directement sous la forme d'une référence "Reference".

10.19 Sauf mention contraire, les différentes possibilités d'affectation "Assignment" sont définies dans les articles suivants de la présente Recommandation | Norme internationale:

<i>Forme d'affectation</i>	<i>Référence de définition</i>
"TypeAssignment"	13.1
"ValueAssignment"	13.2
"ValueSetTypeAssignment"	13.4
"ObjectClassAssignment"	Rec. UIT-T X.681 ISO/CEI 8824-2 paragraphe 9.1
"ObjectAssignment"	Rec. UIT-T X.681 ISO/CEI 8824-2 paragraphe 11.1
"ObjectSetAssignment"	Rec. UIT-T X.681 ISO/CEI 8824-2 paragraphe 12.1
"ParameterizedAssignment"	Rec. UIT-T X.683 ISO/CEI 8824-4 paragraphe 8.1

Le premier symbole de chaque affectation "Assignment" est l'une des possibilités de référencement "Reference", indiquant le nom de la référence à définir. Le même nom de référence ne peut être donné à deux affectations d'une même liste d'affectation "AssignmentList".

11 Référenciation des définitions de types et de valeurs

11.1 Les productions suivantes des types et valeurs définis

```

DefinedType ::=
  Externaltypereference |
  typereference |
  ParameterizedType |
  ParameterizedValueSetType
  
```

DefinedValue ::=
Externalvaluereference |
valuereference |
ParameterizedValue

spécifient les séquences qui seront utilisées pour référencer les définitions de types et de valeurs. Le type identifié par un type paramétré "ParameterizedType" ou un type d'ensemble de valeurs paramétrées "ParameterizedValueSetType" et la valeur identifiée par une valeur paramétrée "ParameterizedValue" sont spécifiés dans la Rec. UIT-T X.683 | ISO/CEI 8824-4.

11.2 En-dehors des cas spécifiés au 10.17, les possibilités "typereference", "valuereference", "ParameterizedType", "ParameterizedValueSetType" et "ParameterizedValue" ne devront être utilisées que si la référence indiquée provient du corps de module "ModuleBody" dans lequel un type ou une valeur est affecté(e) (voir 13.1 et 13.2) à la référence de type "typereference" ou de valeur "valuereference".

11.3 Les références externes de type "Externaltypereference" (ou de valeur "Externalvaluereference") ne seront utilisées que si la référence de type (respectivement de valeur) correspondante:

- a) s'est déjà vue affecter un type (respectivement une valeur) (voir 13.1 et 13.2)
- b) ou est présente dans la section IMPORTS

dans le corps de module "ModuleBody" utilisé pour définir la référence "modulereference" correspondante. Il n'est permis de faire référence à un item de la section IMPORTS d'un autre module que s'il n'existe pas plus d'une occurrence de ce symbole dans la section IMPORTS.

NOTE – Ceci n'empêche pas d'importer dans un module deux symboles identiques définis dans deux autres modules différents. Mais si un même symbole "Symbol" apparaît plus d'une fois dans la section IMPORTS d'un module A, il n'est plus possible de faire référence à ce symbole dans le module A depuis un autre module extérieur.

11.4 Une référence externe ne sera utilisée dans un module que pour désigner un item défini dans un autre module; cette référence est assurée par les productions suivantes:

Externaltypereference ::=
modulereference
","
typereference

Externalvaluereference ::=
modulereference
","
valuereference

NOTE – Des productions supplémentaires de référencement externe ("ExternalClassReference", "ExternalObjectReference" and "ExternalObjectSetReference") sont spécifiées dans la Rec. UIT-T X.681 | ISO/CEI 8824-2.

11.5 Lorsque dans le module importateur, la déclaration d'importation "Imports" est effectuée avec désignation des symboles importés "SymbolsImported", le champ "modulereference" de la référence externe apparaîtra dans le champ "GlobalModuleReference" d'une et une seule production "SymbolsFromModule" de la déclaration "SymbolsImported". Lorsque le module importateur est défini avec une déclaration d'importation "Imports" vide, le champ "modulereference" de la référence externe apparaîtra dans la déclaration de définition "ModuleDefinition" du module extérieur dans lequel la "Reference" invoquée est définie.

12 Notation de prise en charge des références à des composantes ASN.1

12.1 Il sera également nécessaire de se référer formellement à des composantes de types, valeurs, etc., ASN.1 à de multiples fins, par exemple, lorsqu'on écrit un texte, pour identifier un type particulier dans un module ASN.1 donné. Le présent article définit une notation qui peut être utilisée pour effectuer de telles références.

12.2 Cette notation permet d'identifier n'importe quelle composante d'un type ensemble ou séquence (qu'elle soit présente à titre obligatoire ou optionnel dans le type).

NOTE – La Rec. UIT-T X.682 | ISO/CEI 8824-3 spécifie la signification de cette notation dans chacune de ses instances d'utilisation. En particulier, la signification d'une référence à un champ absent dans certaines valeurs du type qui les englobe n'entre pas dans le cadre de la présente Recommandation | Norme internationale.

12.3 Toutes les parties d'une définition de type ASN.1 peuvent être pointées de l'extérieur en utilisant la structure syntaxique de référencement absolue "AbsoluteReference":

```
AbsoluteReference ::= "@" GlobalModuleReference
      "."
      ItemSpec
```

```
ItemSpec ::=
  typereference |
  ItemId "." ComponentId
```

```
ItemId ::= ItemSpec
ComponentId ::=
  identifieur | number | "*"
```

NOTE – La production "AbsoluteReference" n'est pas utilisée dans la présente Recommandation | Norme internationale. Elle a été établie aux fins indiquées au 12.1.

12.4 La référence globale de module "GlobalModuleReference" identifie un module ASN.1 (voir 10.1).

12.5 La référence "typereference" est celle d'un type ASN.1 quelconque défini dans le module identifié par "GlobalModuleReference".

12.6 L'identificateur de composante "ComponentId" de chaque élément "ItemSpec" identifie une composante du type désigné par "ItemId". Il s'agira du dernier identificateur "ComponentId" si la composante identifiée n'est pas un type d'ensemble, de séquence, d'ensemble-de, de séquence-de ou de structure "choix".

12.7 Le champ "identifieur" de "ComponentId" peut être utilisé si l'élément "ItemId" antécédent est du type ensemble ou séquence; cet identificateur doit alors être l'un des identificateurs du type nommé "NamedType" de la liste des types de composantes "ComponentTypeList" de cet ensemble ou séquence. Il peut aussi être utilisé si "ItemId" désigne un type "choix"; il doit alors être l'un des identificateurs de type nommé "NamedType" de la liste de types au choix "AlternativeTypeList" de ce type "choix". Il ne peut pas être utilisé dans d'autres cas.

12.8 Le champ "ComponentId" ne peut recevoir un numéro que si l'élément "ItemId" est un type séquence-de ou ensemble-de. La valeur du numéro identifie l'instance du type donné dans la séquence-de ou l'ensemble-de, le numéro "1" correspondant à la première instance. Le numéro "0" identifie une composante conceptuelle de type entier (non présente explicitement dans le transfert et appelée **nombre d'itérations**) qui indique le nombre d'instances dans la structure séquence-de ou ensemble-de présentes dans la valeur du type contenant.

12.9 Le champ "ComponentId" ne peut recevoir un astérisque "*" que si l'élément "ItemId" est un type séquence-de ou ensemble-de. Toute sémantique associée à l'utilisation de la notation "*" s'applique à toutes les composantes de la structure séquence-de ou ensemble-de.

NOTE – Dans l'exemple suivant:

```
M DEFINITIONS ::= BEGIN
  T ::= SEQUENCE {
    a  BOOLEAN,
    b  SET OF INTEGER
  }
END
```

il est possible de faire référence aux composantes de "T" depuis un texte hors d'un module ASN.1 (ou depuis un commentaire) de la manière suivante par exemple:

si (@M.T.b.0 est impair) alors (@M.T.b.*est un entier impair)

dans laquelle il est déclaré que si le nombre de composantes de "b" est impair, alors toutes les composantes de "b" doivent être impaires.

13 Affectation de types et de valeurs

13.1 Un type sera affecté à une référence de type "typereference" par la notation spécifiée dans la production "TypeAssignment":

```
TypeAssignment ::=
  typereference
  "::="
  Type
```

La référence "typereference" ne devra pas être l'un des mots ASN.1 réservés (voir 9.16).

13.2 Une valeur sera affectée à une référence de valeur "valuereference" par la notation spécifiée dans la production "ValueAssignment":

```
ValueAssignment ::=
    valuereference
    Type
    "::="
    Value
```

La valeur "Value" affectée à la référence "valuereference" sera une notation de valeur correspondant au type défini par le champ "Type" (comme spécifié au 13.3).

13.3 "Value" est une notation de valeur d'un type donné si elle est:

- a) soit une notation de valeur prédéfinie "BuiltinValue" de ce type (voir 14.8);
- b) soit une notation de valeur définie "DefinedValue" compatible avec ce type.

13.4 La notation spécifiée par la production d'affectation de type ensemble de valeurs "ValueSetTypeAssignment" permet d'affecter un ensemble de valeurs à une référence de type "typereference":

```
ValueSetTypeAssignment ::= typereference
    Type
    "::="
    ValueSet
```

Cette notation affecte à "typereference" le type défini comme un sous-type du type "Type", et contenant exactement les valeurs spécifiées ou autorisées par "ValueSet". La référence "typereference" ne sera pas un mot ASN.1 réservé (voir 9.16), et il sera possible d'y faire référence comme à un type. L'ensemble de valeurs "ValueSet" est défini au 13.5.

13.5 Un ensemble de valeurs d'un type donné est spécifié par la notation "ValueSet":

```
ValueSet ::= "{" ElementSetSpec "}"
```

Il comprend toutes les valeurs, au nombre d'une au moins, spécifiées par la spécification d'ensemble d'éléments "ElementSetSpec" (voir article 44).

14 Définition des types et valeurs

14.1 Un type est spécifié par la notation "Type":

```
Type ::= BuiltinType | ReferencedType | ConstrainedType
```

14.2 Les types prédéfinis de l'ASN.1 sont spécifiés par la notation "BuiltinType" définie comme suit:

```
BuiltinType ::=
    BitStringType |
    BooleanType |
    CharacterStringType |
    ChoiceType |
    EmbeddedPDVType |
    EnumeratedType |
    ExternalType |
    InstanceOfType |
    IntegerType |
    NullType |
    ObjectClassFieldType |
    ObjectIdentifierType |
    OctetStringType |
    RealType |
    SequenceType |
    SequenceOfType |
    SetType |
    SetOfType |
    TaggedType
```

Les divers types prédéfinis "BuiltinType" possibles sont définis dans les articles suivants (de la présente Recommandation | Norme internationale sauf indication contraire):

BitStringType	19
BooleanType	15
CharacterStringType	33
ChoiceType	26
EmbeddedPDVType	30
EnumeratedType	17
ExternalType	31
InstanceOfType	Rec. UIT-T X.681 ISO/CEI 8824-2, Annexe C
IntegerType	16
NullType	21
ObjectClassFieldType	Rec. X.UIT-T 681 ISO/CEI 8824-2, paragraphe 14.1
ObjectIdentifierType	29
OctetStringType	20
RealType	18
SequenceType	22
SequenceOfType	23
SetType	24
SetOfType	25
TaggedType	28

14.3 Les types référencés de l'ASN.1 sont spécifiés par la notation "ReferencedType":

```

ReferencedType ::=
    DefinedType           |
    UsefulType           |
    SelectionType       |
    TypeFromObject      |
    ValueSetFromObjects

```

La notation "ReferencedType" offre une autre manière de se référer à un autre type (et en dernier lieu à un type prédéfini). Les différentes notations "ReferencedType" et la manière utilisée pour déterminer le type auquel elles renvoient sont spécifiées aux points suivants (de la présente Recommandation | Norme internationale sauf indication contraire):

DefinedType	11.1
UsefulType	38.1
SelectionType	27
TypeFromObject	Rec. UIT-T X.681 ISO/CEI 8824-2, article 15
ValueSetFromObjects	Rec. UIT-T X.681 ISO/CEI 8824-2, article 15

14.4 Le type contraint "ConstrainedType" est défini à l'article 42.

14.5 La présente Recommandation | Norme internationale spécifie l'utilisation de la notation "NamedType" suivante pour spécifier les composantes des types ensemble, séquence et choix:

```

NamedType ::= identifiant Type

```

14.6 L'identificateur "identifiant" est utilisé pour faire référence sans ambiguïté aux composantes d'un type ensemble, séquence ou choix dans les notations de valeurs et les contraintes relationnelles des composantes (voir la Rec. UIT-T X.682 | ISO/CEI 8824-3). Il ne fait pas partie du type et n'a pas d'effet sur celui-ci.

14.7 Une valeur d'un type donné est spécifiée par la notation "Value":

```

Value ::= BuiltinValue | ReferencedValue | ObjectClassFieldValue

```

NOTE – ObjectClassFieldValue est défini dans la Rec. UIT-T X.681 | ISO/CEI 8824-2, paragraphe 14.6.

14.8 Les valeurs des types prédéfinis de l'ASN.1 peuvent être spécifiées par la notation de valeur prédéfinie "BuiltinValue" définie comme suit:

```

BuiltinValue ::=
    BitStringValue       |
    BooleanValue        |
    CharacterStringValue |
    ChoiceValue         |
    EmbeddedPDVValue    |

```

```

EnumeratedValue |
ExternalValue |
InstanceOfValue |
IntegerValue |
NullValue |
ObjectIdentifierValue |
OctetStringValue |
RealValue |
SequenceValue |
SequenceOfValue |
SetValue |
SetOfValue |
TaggedValue
    
```

Chacune des formes possibles de "BuiltinValue" est définie dans le même article que le type correspondant de la notation "BuiltinType", selon la liste du 14.2 ci-dessus.

14.9 Les valeurs référencées de l'ASN.1 sont spécifiées par la notation "ReferencedValue" suivante:

```

ReferencedValue ::=
    DefinedValue |
    ValueFromObject
    
```

La notation "ReferencedValue" offre une autre manière de se référer à une autre valeur (et en dernier lieu à une valeur prédéfinie). Les différentes notations "ReferencedValue" et la manière utilisée pour déterminer la valeur à laquelle elles renvoient sont spécifiées aux points suivants (de la présente Recommandation | Norme internationale sauf indication contraire):

```

DefinedValue      11.1
ValueFromObject  Rec. UIT-T X.681 | ISO/CEI 8824-2, article 15
    
```

14.10 Indépendamment du fait qu'un type soit prédéfini "BuiltinType", référencé "ReferencedType" ou contraint "ConstrainedType", ses valeurs peuvent être spécifiées par une valeur "BuiltinValue" ou "ReferencedValue" de ce type.

14.11 La valeur d'un type pointé à l'aide de la notation "NamedType" sera définie par la notation "NamedValue":

```

NamedValue ::= identifiant Value
    
```

où l'identificateur "identifiant" est le même que celui de la notation du type nommé "NamedType".

NOTE – L'identificateur "identifiant" fait partie de la notation et non de la valeur proprement dite. Il sert à pointer sans ambiguïté les composantes d'un type ensemble, d'un type séquence ou d'un type choix.

15 Notation du type booléen

15.1 Le type booléen (voir 3.8.7) est déclaré par la notation "BooleanType":

```

BooleanType ::= BOOLEAN
    
```

15.2 L'étiquette des types définis par cette notation est de la classe universelle numéro 1.

15.3 La notation "BooleanValue" permet de définir la valeur d'un type booléen (voir 3.8.55 et 3.8.27):

```

BooleanValue ::= TRUE | FALSE
    
```

16 Notation du type entier (Integer)

16.1 Le type entier (voir 3.8.29) est déclaré par la notation "IntegerType":

```

IntegerType ::=
    INTEGER |
    INTEGER "{" NamedNumberList "}"

NamedNumberList ::=
    NamedNumber |
    NamedNumberList "," NamedNumber
    
```

NamedNumber ::=
identifieur "(" SignedNumber ")" |
identifieur "(" DefinedValue ")"

SignedNumber ::= number | "-" number

16.2 La seconde forme possible de "SignedNumber" ne sera pas utilisée si "number" est nul.

16.3 Dans la déclaration de type, la liste "NamedNumberList" n'est pas significative. Elle est seulement utilisée dans la notation de valeur spécifiée au 16.9.

16.4 La référence de valeur "valuereference" de la valeur définie "DefinedValue" sera du type entier.

NOTE – Un identificateur ne pouvant servir à spécifier la valeur associée à "NamedNumber", la valeur définie "DefinedValue" ne peut jamais être prise pour un entier "IntegerValue". Dans les cas suivants,

a INTEGER ::= 1
T1 ::= INTEGER { a(2) }
T2 ::= INTEGER { a(3), b(a) }
c T2 ::= b
d T2 ::= a

"c" désigne la valeur "1" puisqu'il ne peut être une référence à la deuxième ou troisième occurrence de "a", et "d" désigne la valeur "3".

16.5 Les valeurs des nombres signés "SignedNumber" et des valeurs définies "DefinedValue" apparaissant dans la liste des nombres nommés "NamedNumberList" seront toutes différentes et représentent les valeurs distinctives du type entier.

16.6 Les identificateurs apparaissant dans la liste "NamedNumberList" seront tous différents.

16.7 L'ordre des séquences "NamedNumber" de la liste "NamedNumberList" n'est pas significatif.

16.8 L'étiquette des types définis par cette notation est la classe universelle numéro 2.

16.9 La valeur d'un type entier est déclarée par la notation "IntegerValue":

IntegerValue ::=
SignedNumber |
identifieur

16.10 L'identificateur "identifieur" de la valeur "IntegerValue" sera l'un des identificateurs du type "IntegerType" auquel la valeur est associée, et représentera le nombre correspondant.

NOTE – Pour faire référence à un entier ayant reçu un identificateur, il est préférable d'utiliser l'identificateur "identifieur" de la valeur "IntegerValue".

17 Notation du type énuméré

17.1 Le type énuméré (voir 3.8.24) est déclaré par la notation "EnumeratedType":

EnumeratedType ::=
ENUMERATED "{" Enumeration "}"

Enumeration ::=
EnumerationItem | EnumerationItem "," Enumeration

EnumerationItem ::=
identifieur | NamedNumber

NOTES

1 Chaque valeur d'un type énuméré a un identificateur associé à un entier distinct. Toutefois, les valeurs elles-mêmes ne sont pas censées avoir une sémantique d'entier. Si l'élément énuméré "EnumerationItem" est fourni sous la forme d'un nombre nommé "NamedNumber", il rend possible le contrôle de la représentation de la valeur pour faciliter des extensions compatibles.

2 Les valeurs numériques dans les nombres nommés "NamedNumber" d'une séquence "Enumeration" ne sont pas nécessairement ordonnées ni consécutives.

17.2 L'identificateur et le nombre signé "SignedNumber" des différents nombres "NamedNumber" seront différents de tous les autres identificateurs et nombres signés de la séquence "Enumeration". Les dispositions des 16.2 et 16.4 s'appliquent aussi à chaque nombre nommé "NamedNumber".

17.3 S'il s'agit d'un identificateur "identifier", chaque item "EnumerationItem" d'un type énuméré "EnumeratedType" se voit affecter un entier non négatif distinct. A cette fin, on affecte successivement les entiers en commençant par 0, à l'exception de ceux qui sont utilisés dans les items "EnumerationItem" comme nombres nommés "NamedNumber".

NOTE – Un entier est associé à l'item "EnumerationItem" pour aider à en définir les règles de codage. Mais cet artifice n'est pas utilisé ailleurs dans la spécification de la notation ASN.1.

17.4 L'étiquette du type énuméré est de la classe universelle, numéro 10.

17.5 La valeur d'un type énuméré est déclarée par la notation "EnumeratedValue":

EnumeratedValue ::= identifier

17.6 L'identificateur "identifier" d'une valeur "EnumeratedValue" sera identique à l'identificateur présent dans la séquence "EnumeratedType" et auquel la valeur est associée.

18 Notation du type réel

18.1 Le type réel (voir 3.8.41) est déclaré par la notation "RealType":

RealType ::= REAL

18.2 L'étiquette du type réel est de la classe universelle numéro 9.

18.3 Les valeurs du type réel sont les valeurs PLUS-INFINITY et MINUS-INFINITY, ainsi que les nombres réels qui peuvent être représentés par la formule suivante faisant intervenir trois entiers M, B et E:

$$M \times B^E$$

où M est la mantisse, B la base et E l'exposant.

18.4 Le type réel possède un type associé qui sert à préciser la définition des valeurs abstraites de type réel et qui sert également à prendre en charge les notations de valeurs et de sous-types du type réel.

NOTE – Les règles de codage peuvent définir un type différent servant à spécifier les codages; ils peuvent aussi spécifier les codages sans faire référence au type associé. En particulier, les règles de codage de base BER et compact PER assurent le codage décimal codé binaire (BCD) si la base est égale à 10, et un codage permettant une transformation efficace vers et depuis les représentations en virgule flottante sur matériel informatique si la base est égale à 2.

18.5 Le type associé servant à la définition des valeurs et au sous-typage est le suivant (les commentaires ont force de norme):

```
SEQUENCE {
    mantissa INTEGER,
    base INTEGER (2|10),
    exponent INTEGER
    -- La valeur réelle associée est égale à "mantissa" fois "base" puissance "exponent"
}
```

NOTES

1 Les valeurs représentées en base 2 et en base 10 sont considérées comme des valeurs abstraites distinctes même si elles correspondent à la même valeur réelle, et peuvent recouvrir des sémantiques d'application différentes.

2 La notation "REAL (WITH COMPONENTS { ... , base (10)})" peut être utilisée pour restreindre l'ensemble des valeurs aux valeurs abstraites en base 10 (ou de manière similaire aux valeurs abstraites de base 2)

3 Ce type peut véhiculer une représentation finie exacte de tout nombre pouvant être enregistré sur un matériel standard en virgule flottante, ainsi que tout nombre à représentation en caractères décimaux finie.

18.6 La valeur d'un type réel sera déclarée au moyen de la notation "RealValue":

RealValue ::=
NumericRealValue | SpecialRealValue

NumericRealValue ::= 0 |
SequenceValue -- valeur associée du type séquence

SpecialRealValue ::=
PLUS-INFINITY | MINUS-INFINITY

Si la valeur est nulle, elle sera représentée par la forme "0" et la seconde forme possible de "NumericRealValue" ne sera pas utilisée.

19 Notation du type chaîne binaire (bitstring)

19.1 Le type chaîne binaire (voir 3.8.6) sera déclaré au moyen de la notation "BitstringType":

```
BitStringType ::=
    BIT STRING
    BIT STRING "{" NamedBitList "}"
```

```
NamedBitList ::=
    NamedBit      |
    NamedBitList "," NamedBit
```

```
NamedBit ::=
    identifieur "(" number ")" |
    identifieur "(" DefinedValue ")"
```

19.2 Dans une chaîne binaire, le premier bit est le **bit zéro** et le dernier bit est appelé le **bit de fin**.

NOTE – Cette terminologie est utilisée pour spécifier la notation des valeurs et définir les règles de codage.

19.3 "DefinedValue" pointera vers une valeur non négative de type entier.

19.4 Les valeurs "number" ou "DefinedValue" de la liste "NamedBitList" seront toutes différentes, et correspondront chacune au numéro d'un bit distinct dans une chaîne binaire.

19.5 Les identificateurs "identifieur" apparaissant dans la liste "NamedBitList" seront tous différents.

NOTES

1 L'ordre des productions "NamedBit" dans la liste "NamedBitList" n'est pas significatif.

2 Etant donné qu'un identificateur de la liste "NamedBitList" ne peut pas servir à spécifier la valeur associée à un bit nommé "NamedBit", la valeur "DefinedValue" ne peut jamais être interprétée comme étant une valeur d'entier "IntegerValue". Par conséquent, dans la production suivante,

```
a INTEGER ::= 1
T1 ::= INTEGER { a(2) }
T2 ::= BIT STRING { a(3), b(a) }
```

la dernière occurrence de "a" représente la valeur 1, puisqu'elle ne peut correspondre ni à la deuxième ni à la troisième occurrence de "a".

19.6 La présence d'une liste "NamedBitList" n'a aucun effet sur l'ensemble des valeurs abstraites du type déclaré: les valeurs pourront comporter des bits de valeur 1 autres que les bits nommés.

19.7 Lorsque une liste de bits nommés "NamedBitList" est utilisée dans la définition d'un type de chaîne binaire, les règles de codage ASN.1 peuvent ajouter (ou supprimer) un nombre arbitraire de bits de fin nuls aux valeurs en cours de codage ou de décodage. Les concepteurs d'applications doivent donc s'assurer que des valeurs ne différant que par le nombre de "0" en bout de chaîne ne sont pas sémantiquement différentes.

19.8 Ce type porte l'étiquette de classe universelle numéro 3.

19.9 Une valeur d'un type chaîne binaire "bitstring" est déclarée au moyen de la notation "BitStringValue":

```
BitStringValue ::=
    bstring          |
    hstring          |
    "{" IdentifieurList "}" |
    "{" "}"
```

```
IdentifieurList ::=
    identifieur      |
    IdentifieurList "," identifieur
```

19.10 Chaque identificateur "identifieur" de la notation "BitStringValue" sera identique à l'identificateur de la production "BitStringType" auquel cette valeur est associée.

19.11 La notation "BitStringValue" représente une valeur de chaîne binaire dont les bits de numéros correspondant aux identificateurs contiennent des "1", tous les autres bits étant à "0".

NOTE – La séquence de production "{" "}" sert à représenter une chaîne binaire ne contenant pas de "1".

19.12 Lors de la spécification de règles de codage pour les chaînes binaires, on utilisera les expressions **premier bit** et **dernier bit** dans la désignation des bits, le premier bit portant le numéro zéro (voir 19.2).

19.13 Dans la notation "bstring", le **premier bit** est à gauche et le **dernier bit** à droite.

19.14 Dans la notation "hstring", le bit le plus significatif de chaque chiffre hexadécimal est celui de gauche.

NOTE – Cette notation n'impose aucune contrainte quant à la manière dont les règles de codage convertissent une chaîne binaire en octets pour le transfert.

19.15 La notation "hstring" ne sera utilisée que si la valeur de chaîne binaire comprend un nombre de bits multiple de quatre.

EXEMPLE

'A98A'H

et

'1010100110001010'B

sont les deux notations possibles d'une même valeur de chaîne binaire. Si le type a été défini à l'aide d'une liste "NamedBitList", le bit de fin, qui est nul, ne fait pas partie de la valeur, dont la longueur est donc de 15 bits. Si le type a été défini sans liste "NamedBitList", le bit de fin nul fait partie de la valeur, qui compte alors 16 bits.

20 Notation du type chaîne d'octets (octetstring)

20.1 Le type chaîne d'octet "octetstring" (voir 3.8.37) est déclaré par la notation "OctetStringType":

OctetStringType ::= OCTET STRING

20.2 Ce type porte l'étiquette de classe universelle numéro 4.

20.3 Une valeur d'un type chaîne d'octet est déclarée par la notation "OctetStringValue":

OctetStringValue ::=
bstring |
hstring

20.4 Lors de la spécification des règles de codage d'une chaîne d'octets, les octets extrêmes sont désignés par les expressions **premier octet** et **dernier octet**, et les bits extrêmes d'un octet sont désignés par les expressions **bit de plus fort poids** et **bit de plus faible poids**.

20.5 Si la valeur est déclarée au moyen de la notation "bstring", le bit le plus à gauche sera le bit de plus fort poids du premier octet. Si le nombre de bits de la chaîne binaire n'est pas un multiple de huit, elle sera interprétée comme se terminant implicitement par des bits complémentaires tous nuls, la complétant au prochain multiple de huit bits.

20.6 Si la valeur est déclarée au moyen de la notation "hstring", le chiffre hexadécimal le plus à gauche sera le demi-octet le plus significatif du premier octet.

20.7 Si le nombre de chiffres de la chaîne hexadécimale n'est pas paire, elle sera interprétée comme se terminant implicitement par un chiffre hexadécimal complémentaire nul.

21 Notation du type néant (Null)

21.1 Le type néant "Null" (voir 3.8.32) est déclaré au moyen de la notation "NullType":

NullType ::= NULL

21.2 Ce type porte l'étiquette de classe universelle numéro 5.

21.3 Une valeur du type néant est déclarée par la notation "NullValue":

NullValue ::= NULL

22 Notation des types séquence

22.1 Un type séquence (voir 3.8.45) est déclaré au moyen de la notation "SequenceType":

```
SequenceType ::=
    SEQUENCE "{" ComponentTypeList "}" |
    SEQUENCE "{" "}"

ComponentTypeList ::=
    ComponentType |
    ComponentTypeList "," ComponentType

ComponentType ::=
    NamedType |
    NamedType OPTIONAL |
    NamedType DEFAULT Value |
    COMPONENTS OF Type
```

22.2 Lorsque la production "ComponentTypeList" apparaît à l'intérieur de la définition d'un module pour lequel a été choisi l'étiquetage automatique (voir 10.3), et qu'aucune des occurrences de "NamedType" correspondant aux trois premières formes possibles de déclaration de "ComponentType" ne contient de type étiqueté "TaggedType", alors la transformation d'étiquetage automatique est adoptée pour l'ensemble de la liste des types de composantes "ComponentTypeList"; sinon, l'étiquetage automatique n'est pas appliqué.

NOTES

1 L'utilisation de la notation de type étiqueté dans la définition de la liste des composantes d'un type séquence permet au spécificateur de garder le contrôle de l'étiquetage, contrairement à ce qui se passe avec l'affectation par un mécanisme d'étiquetage automatique. Ainsi, dans le cas suivant:

```
T ::= SEQUENCE { a INTEGER, b [1] BOOLEAN, c OCTET STRING }
```

aucun étiquetage automatique n'est appliqué à la liste des composantes a, b, c, même si cette définition du type de séquence T apparaît à l'intérieur d'un module pour lequel l'étiquetage automatique a été sélectionné.

2 Seules les occurrences de productions "ComponentTypeList" apparaissant à l'intérieur d'un module pour lequel l'étiquetage automatique a été choisi sont des candidats possibles pour une transformation par étiquetage automatique.

22.3 La décision d'appliquer l'étiquetage automatique est prise individuellement pour chaque occurrence de la liste "ComponentTypeList" *avant* la transformation COMPONENTS OF spécifiée au 22.4. Toutefois, comme l'indique 22.6, l'étiquetage automatique, s'il s'applique, est appliqué *après* la transformation COMPONENTS OF.

NOTE – L'effet de cette disposition est que l'étiquetage automatique n'a pas lieu là où des étiquettes explicites sont présentes dans la liste "ComponentTypeList", mais qu'il a lieu même si des étiquettes sont présentes dans le type déclaré à la suite d'une transformation "COMPONENTS OF".

22.4 Le type indiqué dans la quatrième forme possible de "ComponentType" doit être un type séquence. La notation "COMPONENTS OF Type" sera utilisée pour définir l'inclusion, à cet endroit de la liste "ComponentTypeList", de tous les types de composantes apparaissant dans le type désigné.

NOTE – Cette transformation est logiquement effectuée avant l'exécution des dispositions des points suivants.

22.5 Lorsqu'une ou plusieurs occurrences du type "ComponentType" marqués par la caractéristique OPTIONAL ou DEFAULT se suivent, leurs étiquettes ainsi que celles de tout type de composante placée immédiatement à la suite doivent être distinctes (voir l'article 28). Si l'étiquetage automatique a été sélectionné, cette spécification d'étiquettes distinctes ne s'applique qu'après étiquetage automatique, et devra alors toujours être satisfaite.

22.6 L'étiquetage automatique d'une occurrence de la liste "ComponentTypeList" est logiquement effectué *après* la transformation spécifiée au 22.4, mais seulement si 22.2 établit qu'il s'applique bien à cette occurrence. L'étiquetage automatique agit sur chaque type "ComponentType" de la liste "ComponentTypeList" en remplaçant le "Type" déclaré à l'origine dans la production "NamedType" par une occurrence "TaggedType" telle que celle-ci est spécifiée au 22.7.

22.7 Le type étiqueté de remplacement "TaggedType" est spécifié comme suit:

- la notation "TaggedType" de remplacement utilise la forme de production "Tag Type";
- la classe "Class" du type de remplacement "TaggedType" est vide (c'est-à-dire que l'étiquetage est propre au contexte);
- le numéro de classe "ClassNumber" du type de remplacement "TaggedType" est égal à 0 pour la première composante de type, à 1 pour la deuxième, et s'incrémente ainsi de suite à chaque numéro d'étiquette;
- le "Type" du type de remplacement "TaggedType" est le "Type" d'origine remplacé.

NOTES

1 Le paragraphe 28.6 indique les règles qui régissent la spécification de l'étiquetage implicite ou explicite pour les types étiquetés de remplacement "TaggedType". L'étiquetage automatique est toujours implicite à moins que le "Type" soit une notation de type choix ou de type ouvert, ou une référence muette "DummyReference" (voir 8.4 de la Rec. UIT-T X.683 | ISO/CEI 8824-4), auquel cas l'étiquetage est explicite.

2 Une fois 22.6 exécuté, les étiquettes des différentes composantes sont complètement déterminées, et ne sont plus modifiées même lorsque le type séquence est indiqué en référence dans la définition d'une composante dans une autre liste "ComponentTypeList" à laquelle l'étiquetage automatique s'applique. Ainsi, dans le cas suivant:

T ::= SEQUENCE { a Ta, b Tb, c Tc }

E ::= SEQUENCE { f1 E1, f2 T, f3 E3 }

les étiquettes attachées à a, b et c ne sont pas affectées par l'étiquetage automatique éventuellement appliqué aux composantes de E.

3 Lorsqu'un type séquence est le "Type" indiqué dans une structure "COMPONENTS OF Type", chaque occurrence de type "ComponentType" qui y apparaît est dupliquée par application du § 22.4 avant l'étiquetage automatique éventuel du type séquence qui y fait référence. Ainsi, dans le cas suivant:

T ::= SEQUENCE { a Ta, b SEQUENCE { b1 T1, b2 T2, b3 T3 }, c Tc }

W ::= SEQUENCE { x Wx, COMPONENTS OF T, y Wy }

les étiquettes de a, b et c dans l'expression de T ne sont pas nécessairement identiques aux étiquettes de a, b et c dans l'expression de W si W a été défini dans un environnement d'étiquetage automatique, mais les étiquettes de b1, b2 et b3 sont les mêmes dans T et W. En d'autres termes, l'étiquetage automatique n'est appliqué qu'une seule fois à une liste "ComponentTypeList" donnée.

4 Le sous-typage n'a aucun impact sur l'étiquetage automatique.

5 En étiquetage automatique, l'insertion de nouvelles composantes peut induire des modifications sur les autres composantes par suite des effets collatéraux de la modification des étiquettes.

22.8 Si une production comporte une déclaration "OPTIONAL" ou "DEFAULT", la valeur correspondante peut être omise dans la valeur du nouveau type.

22.9 Si une production comporte une déclaration "DEFAULT", l'omission de la valeur correspondante est équivalente à l'insertion de la valeur définie par "Value", qui doit être la notation d'une valeur du type défini par "Type" dans la séquence de production "NamedType".

22.10 Les identificateurs "identifier" des séquences de production "NamedType" de la liste "ComponentTypeList" (plus les identificateurs obtenus par le développement des expressions COMPONENTS OF) seront tous distincts.

22.11 Tous les types séquence porteront l'étiquette de la classe universelle numéro 16.

NOTE – Les types séquence-de portent la même étiquette que les types séquence (voir 23.2).

22.12 Une valeur de type séquence sera définie par la notation "SequenceValue":

SequenceValue ::=

"{" ComponentValueList "}" |
"{" "}"

ComponentValueList ::=

NamedValue |
ComponentValueList "," NamedValue

22.13 La notation "{ }" ne sera utilisée que si:

- a) toutes les séquences "ComponentType" de la production "SequenceType" comportent la déclaration "DEFAULT" ou "OPTIONAL", et toutes les valeurs sont omises; ou
- b) la notation du type était "SEQUENCE{ }".

22.14 Une valeur "NamedValue" figurera pour chaque type "NamedType" du type "SequenceType" ne comportant pas la déclaration OPTIONAL ou DEFAULT, et ces valeurs apparaîtront dans le même ordre que les types "NamedType" de la séquence correspondante.

23 Notation des types séquence-de

23.1 Un type séquence-de (voir 3.8.46) est défini à partir d'un "Type" donné par la notation "SequenceOfType":

SequenceOfType ::= SEQUENCE OF Type

23.2 Les types séquence-de portent tous l'étiquette classe universelle numéro 16.

NOTE – Les types séquence portent la même étiquette que les types séquence-de (voir 22.11).

23.3 Une valeur de type séquence-de est déclarée par la notation "SequenceOfValue":

SequenceOfValue ::= "{" ValueList "}" | "{" "}"

ValueList ::=

Value |
ValueList "," Value

La notation "{" "}" est utilisée quand la valeur séquence-de est une liste vide.

23.4 Chaque valeur "Value" de la liste "ValueList" sera du type spécifié dans la déclaration "SequenceOfType".

NOTE – L'ordre de ces valeurs peut avoir une signification sémantique.

24 Notation des types ensemble

24.1 Un type ensemble (voir 3.8.47) est défini à partir d'autres types par la notation "SetType":

SetType ::=

SET "{" ComponentTypeList "}" |
SET "{" "}"

La liste de types de composantes "ComponentTypeList" est spécifiée au 22.1

24.2 Le "Type" indiqué dans la quatrième forme possible de la notation "ComponentType" (voir 22.1) sera un type ensemble. La notation "COMPONENTS OF Type" sera utilisée à cet endroit de la liste "ComponentTypeList" pour définir l'inclusion de toutes les séquences "ComponentType" figurant dans le type désigné.

NOTE – Cette transformation doit être logiquement effectuée avant d'appliquer les dispositions des points suivants.

24.3 Les types "ComponentType" d'un type ensemble porteront tous des étiquettes différentes (voir l'article 28).

NOTE – Lorsque l'étiquetage par défaut "TagDefault" du module dans lequel cette notation apparaît est "AUTOMATIC TAGS", il résulte des dispositions du 22.6 que l'étiquetage automatique est appliqué quelque soit le type "ComponentType" effectif.

24.4 Les paragraphes 22.2 et 22.6 à 22.10 s'appliquent également aux types ensemble.

24.5 Les types ensemble portent tous l'étiquette de classe universelle numéro 17.

NOTE – Les types ensemble-de portent la même étiquette que les types ensemble (voir 25.2).

24.6 Dans un type ensemble, l'ordre des valeurs n'est porteur d'aucun contenu sémantique.

24.7 Une valeur de type ensemble est déclarée par la notation "SetValue":

SetValue ::= "{" ComponentValueList "}" | "{" "}"

La liste de valeurs de composantes "ComponentValueList" est spécifiée au 22.12.

24.8 La liste "SetValue" ne se réduira à l'ensemble vide "{" "}" que si:

a) toutes les séquences "ComponentType" de la production "SetType" comportent la déclaration "DEFAULT" ou "OPTIONAL", et toutes les valeurs sont omises; ou

b) la notation du type était "SET{ }".

24.9 Une valeur "NamedValue" figurera pour chaque type "NamedType" du type "SetType" ne comportant pas la déclaration "OPTIONAL" ou "DEFAULT".

NOTE – Ces valeurs "NamedValue" peuvent figurer dans n'importe quel ordre.

25 Notation des types ensemble-de

25.1 Un type ensemble-de (voir 3.8.48) est défini à partir d'autres types par la notation "SetOfType":

SetOfType ::= SET OF Type

25.2 Les types ensemble-de portent tous l'étiquette de classe universelle numéro 17.

NOTE – Les types ensemble portent la même étiquette que les types ensemble-de (voir 24.5).

25.3 Une valeur de type ensemble-de est déclarée par la notation "SetOfValue":

SetOfValue ::= "{" ValueList "}" | "{" "}"

La liste de valeurs "ValueList" est spécifiée au 23.3.

La notation "{" "}" est utilisée quand la valeur ensemble-de est une liste vide.

25.4 Chaque valeur "Value" de la liste "ValueList" sera du type spécifié dans la déclaration "SetOfType".

NOTES

1 L'ordre des valeurs n'est porteur d'aucun contenu sémantique.

2 Il n'est pas demandé aux règles de codage de préserver l'ordre de ces valeurs.

3 Le type ensemble-de n'est pas un ensemble de valeurs au sens mathématique. Ainsi, pour un ENSEMBLE-D'ENTRIERS, les valeurs "{ 1 }" et "{ 1 1 }" sont distinctes.

26 Notation des types choix

26.1 Un type choix (voir 3.8.13) est défini à partir d'autres types par la notation "ChoiceType":

ChoiceType ::= CHOICE "{" AlternativeTypeList "}"

AlternativeTypeList ::=

NamedType |
AlternativeTypeList "," NamedType

NOTE – T ::= CHOICE { a A } et A ne sont pas des types identiques, et peuvent être codés différemment par les règles de codage.

26.2 Les types définis dans la liste "AlternativeTypeList" porteront des étiquettes distinctes (voir l'article 28).

NOTE – Lorsque l'étiquetage par défaut "TagDefault" du module dans lequel cette notation apparaît est "AUTOMATIC TAGS", il résulte des dispositions du 22.6 que les étiquettes générées seront distinctes.

26.3 Lorsque la production "AlternativeTypeList" apparaît dans la définition d'un module pour lequel l'étiquetage automatique a été choisi (voir 10.3), et si aucune des occurrences "NamedType" de cette liste ne contient un "Type" étiqueté par une production "TaggedType", alors l'étiquetage automatique est adopté pour l'ensemble de la liste "AlternativeTypeList"; sinon, l'étiquetage automatique n'est pas appliqué. Lorsqu'il est adopté, l'étiquetage automatique d'une liste "AlternativeTypeList" est appliqué à chaque type nommé "NamedType" de la liste en remplaçant le "Type" déclaré à l'origine dans la production "NamedType" par une occurrence "TaggedType" telle que celle-ci est spécifiée au 22.7.

26.4 Le type choix contient des valeurs qui ne portent pas tous la même étiquette. (L'étiquette dépend du choix effectué pour la générer la valeur du type choix.)

26.5 Lorsque le type choix est utilisé dans un cas où la présente Recommandation | Norme internationale impose aux types de porter des étiquettes distinctes (voir 22.5, 24.3 et 26.2), la disposition s'appliquera à toutes les étiquettes possibles des valeurs du type choix.

Les exemples suivants, dans lesquels on suppose que l'étiquetage par défaut "TagDefault" n'est pas l'étiquetage automatique "AUTOMATIC TAGS", illustrent cette spécification.

EXEMPLES

```

1      A ::= CHOICE
          {b  B,
           c  NULL}

      B ::= CHOICE
          {d  [0] NULL,
           e  [1] NULL}

2      A ::= CHOICE
          {b  B,
           c  C}

      B ::= CHOICE
          {d  [0] NULL,
           e  [1] NULL}

      C ::= CHOICE
          {f  [2] NULL,
           g  [3] NULL}
    
```

3 (INCORRECT)

```

A ::= CHOICE
    {b    B,
     c    C}

B ::= CHOICE
    {d    [0] NULL,
     e    [1] NULL}

C ::= CHOICE
    {f    [0] NULL,
     g    [1] NULL}

```

Les exemples 1 et 2 correspondent à une utilisation correcte de la notation. L'exemple 3 n'est pas correct en l'absence d'étiquetage automatique, car les étiquettes des types d et f d'une part, et e et g d'autre part, sont identiques.

26.6 Les identificateurs "identifier" de toutes les productions "NamedType" d'une même liste "AlternativeTypeList" seront tous distincts.

26.7 Une valeur de type choix est déclarée par la notation "ChoiceValue":

```
ChoiceValue ::= identifier ":" Value
```

26.8 "Value" sera la notation d'une valeur du type de la liste "AlternativeTypeList" désigné par "identifier".

27 Notation des types sélection

27.1 Un type sélection (voir 3.8.44) est défini par la notation "SelectionType":

```
SelectionType ::= identifier "<" Type
```

où "Type" désigne un type choix, et où "identifier" est l'identificateur d'une production "NamedType" figurant dans la liste "AlternativeTypeList" de ce type choix.

27.2 Lorsque le type "SelectionType" est utilisé comme type nommé "NamedType", son identificateur "identifier" sera celui de ce type nommé.

27.3 Lorsque le type "SelectionType" est utilisé comme type "Type", son identificateur "identifier" est retenu et le type désigné est celui de la forme sélectionnée.

27.4 La notation pour une valeur de type sélection sera la notation d'une valeur du type désigné par le type "SelectionType".

28 Notation des types étiquetés

Un type étiqueté "TaggedType" (voir 3.8.53) est un type nouveau, isomorphe d'un type existant, mais portant une étiquette différente. Le type étiqueté est principalement utilisé là où la présente Recommandation | Norme internationale spécifie l'emploi de types portant des étiquettes distinctes (voir 22.5, 24.3, 26.2 et 26.4). L'adoption de l'étiquetage automatique AUTOMATIC TAGS comme étiquetage par défaut "TagDefault" dans un module permet d'accomplir un tel étiquetage sans que la notation de type étiqueté n'apparaisse explicitement dans le module.

NOTE – Lorsqu'un protocole détermine qu'à un moment donné des valeurs de différents types de données peuvent être transmises, des étiquettes distinctes peuvent s'avérer nécessaires pour permettre au destinataire de les décoder correctement.

28.1 Un type étiqueté est défini par la notation "TaggedType":

```

TaggedType ::=
    Tag Type           |
    Tag IMPLICIT Type |
    Tag EXPLICIT Type

```

```
Tag ::= "[" Class ClassNumber "]"
```

```

ClassNumber ::=
    number |
    DefinedValue

```

```
Class ::=
    UNIVERSAL      |
    APPLICATION    |
    PRIVATE        |
    empty
```

28.2 La référence "valuereference" de la valeur "DefinedValue" sera du type entier et se verra affecter une valeur non négative.

28.3 Le type nouveau est isomorphe de l'ancien, mais porte une étiquette de la classe "Class" et de numéro "ClassNumber", à moins que la classe ne soit vide (option "empty"), ce qui correspond à une classe spécifique au contexte, auquel cas l'étiquette porte seulement un numéro "ClassNumber".

28.4 Seuls les types définis dans la présente Recommandation | Norme internationale peuvent porter une étiquette de la classe universelle "UNIVERSAL".

NOTES

- 1 L'utilisation d'étiquettes de la classe universelle fait l'objet d'accords périodiques entre l'ISO et l'UIT -T.
- 2 L'Annexe F.2.12 comporte des directives et des conseils stylistiques relatifs à l'utilisation des classes d'étiquettes.

28.5 L'étiquetage est toujours implicite ou explicite. L'étiquetage implicite indique, pour les règles de codage qui en offrent le choix, qu'il n'est pas nécessaire d'identifier explicitement en transfert l'étiquette d'origine du "Type" dans le type étiqueté.

NOTE – Il peut être utile de conserver l'ancienne étiquette si celle-ci est de classe universelle et qu'elle identifie donc de façon non ambiguë l'ancien type sans connaître la définition en notation ASN.1 du nouveau type. Toutefois, l'utilisation de l'étiquetage implicite permet de minimiser le nombre d'octets à transférer. Un exemple de codage utilisant la déclaration d'étiquetage IMPLICIT est donné dans la Rec. UIT-T X.690 | ISO/CEI 8825-1.

28.6 La structure d'étiquetage spécifie un étiquetage explicite si l'une des conditions suivantes est vérifiée:

- a) la forme "Tag EXPLICIT Type" est utilisée;
- b) la forme "Tag Type" est utilisée et l'étiquetage par défaut "TagDefault" du module est explicite "EXPLICIT TAGS" ou vide "empty";
- c) la forme "Tag Type" est utilisée et l'étiquetage par défaut "TagDefault" du module est implicite "IMPLICIT TAGS" ou automatique "AUTOMATIC TAGS", mais le type défini par "Type" est un type choix, un type ouvert ou une référence muette "DummyReference" (voir 8.3 de la Rec. UIT-T X.683 | ISO/CEI 8824-4).

Dans tous les autres cas, la structure d'étiquetage spécifie un étiquetage implicite.

28.7 Si la classe indiquée "Class" est vide (option "empty"), l'utilisation de l'étiquette "Tag" n'est soumise à aucune autre restriction que l'obligation spécifiée par 22.5, 24.3 et 26.2 imposant d'avoir des étiquettes distinctes.

28.8 La déclaration d'étiquetage implicite "IMPLICIT" ne sera pas utilisée si le "Type" indiqué est un type choix, un type ouvert ou une référence muette "DummyReference" (voir 8.3 de la Rec. UIT-T X.683 | ISO/CEI 8824-4).

28.9 Une valeur de type "TaggedType" est déclarée par la notation "TaggedValue":

```
TaggedValue ::= Value
```

où "Value" est la notation d'une valeur du type "Type" dans la notation "TaggedType".

NOTE – L'étiquette "Tag" n'apparaît pas dans cette notation.

29 Notation du type identificateur d'objet

29.1 Le type identificateur d'objet (voir 3.8.36) est déclaré au moyen de la notation "ObjectIdentifierType":

```
ObjectIdentifierType ::= OBJECT IDENTIFIER
```

29.2 Ce type porte l'étiquette de classe universelle numéro 6.

29.3 Une valeur d'identificateur d'objet est déclarée par la notation "ObjectIdentifierValue":

```
ObjectIdentifierValue ::=
    "{" ObjIdComponentList "}" |
    "{" DefinedValue ObjIdComponentList "}"
```

```

ObjIdComponentList ::=
    ObjIdComponent |
    ObjIdComponent ObjIdComponentList

ObjIdComponent ::=
    NameForm |
    NumberForm |
    NameAndNumberForm

NameForm ::= identifiant

NumberForm ::= number | DefinedValue

NameAndNumberForm ::=
    identifiant "(" NumberForm ")"

```

29.4 La référence "valuereference" de la valeur "DefinedValue" de la notation "NumberForm" sera du type entier et se verra affecter une valeur non négative.

29.5 La référence "valuereference" de la valeur "DefinedValue" de la notation "ObjectIdentifierValue" sera du type identificateur d'objet.

29.6 La notation "NameForm" ne sera utilisée que pour les composantes d'identificateur d'objet dont la valeur numérique et l'identificateur sont spécifiés dans les Annexes B à D, et sera l'un de ces identificateurs.

29.7 Le numéro "number" de la notation "NumberForm" sera la valeur numérique affectée à la composante d'identificateur d'objet.

29.8 Le champ "identifiant" de la notation "NameAndNumberForm" sera spécifié quand une valeur numérique est affectée à la composante d'identificateur d'objet.

NOTE – Les autorités affectant des valeurs numériques aux composantes d'identificateur d'objet sont identifiées dans les annexes de la présente Recommandation | Norme internationale.

29.9 La sémantique d'une valeur d'identificateur d'objet est définie par référence à un **arbre d'identificateurs d'objets**. Un arbre d'identificateurs d'objets est un arbre dont la racine correspond à la présente Recommandation | Norme internationale, et chaque nœud à une autorité administrative responsable de l'affectation des arcs issus de ce nœud. Chaque arc est étiqueté par une composante d'identificateur d'objet qui est une valeur numérique. Chaque objet à identifier est affecté à un nœud et un seul (normalement une feuille) et aucun autre objet, de type identique ou différent, n'est affecté à ce nœud. Un objet est ainsi identifié d'une façon unique et non ambiguë par la séquence des valeurs numériques (composantes de l'identificateur d'objet) étiquetant les arcs du chemin allant de la racine au nœud attribué à cet objet.

NOTE – Les valeurs d'identificateur d'objet contiennent au moins deux composantes d'identificateur d'objet, comme cela est spécifié dans les Annexes B à D.

29.10 Sémantiquement, une valeur d'identificateur d'objet est une liste ordonnée de valeurs de composantes d'identificateur d'objet. Chaque valeur de composante d'identificateur d'objet identifie un arc de l'arbre des identificateurs d'objets à partir de la racine. La dernière valeur de composante d'identificateur d'objet identifie un arc allant à un nœud auquel un objet a été affecté. C'est cet objet qui est identifié par la valeur d'identificateur d'objet. La partie significative de la composante d'identificateur d'objet est la forme "NameForm" ou "NumberForm" à laquelle il se ramène, et qui donne la valeur numérique de cette composante d'identificateur d'objet.

NOTE – En général, un objet est une classe informationnelle (par exemple un format de fichier) plutôt qu'une instance d'une telle classe (par exemple un fichier particulier). C'est donc à la classe informationnelle (définie par une spécification quelconque à laquelle il peut être fait référence) plutôt qu'à l'élément d'information lui-même qu'est affecté un emplacement dans l'arbre.

29.11 Quand la notation "ObjectIdentifierValue" comprend une valeur "DefinedValue", la liste des composantes d'identificateur d'objet à laquelle elle se réfère s'ajoute par préfixation aux composantes figurant explicitement dans la valeur.

NOTE – Chaque fois qu'une Recommandation | Norme internationale ou un autre document affecte des valeurs du type OBJECT IDENTIFIER à des objets, il est recommandé d'établir une annexe ou un appendice résumant les affectations qui y sont faites. Il est également recommandé que l'autorité qui affecte une valeur du type OBJECT IDENTIFIER à un objet lui affecte également une valeur du type descripteur d'objet "ObjectDescriptor" (voir l'article 41).

EXEMPLES

Avec les identificateurs affectés selon les dispositions de l'Annexe B, les valeurs:

{ iso standard 8571 pci (1) }

et

{ 1 0 8571 1 }

identifient tous deux un même objet, "pci", défini dans ISO 8571.

Avec la définition additionnelle suivante:

ftam OBJECT IDENTIFIER ::= { iso standard 8571 }

la valeur suivante est équivalente aux deux premières valeurs:

{ ftam pci(1) }

30 Notation du type pdv encapsulé

30.1 Un type valeur de donnée de présentation (pdv) encapsulée (voir 3.8.21) est défini par la notation "EmbeddedPDVType":

EmbeddedPDVType ::= EMBEDDED PDV

30.2 Ce type porte l'étiquette de classe universelle numéro 11.

NOTE – Lorsque la négociation de la couche Présentation est utilisée, le type EMBEDDED PDV assure la même fonction que le type EXTERNAL (plus des fonctions additionnelles), mais le train binaire transmis sera différent. Il est recommandé dans ce cas de remplacer dans les prochaines versions des protocoles applicatifs le type EXTERNAL par le type choix CHOICE{external EXTERNAL, embedded-pdv EMBEDDED PDV}. L'Annexe C de la Rec. UIT-T X.681 | ISO/CEI 8824-2 propose de remplacer similairement l'utilisation du type EXTERNAL dans des cas supplémentaires où la négociation de la couche de présentation n'est pas utilisée.

30.3 Ce type regroupe des valeurs constituées:

- a) du codage d'une valeur de donnée simple, pouvant être une valeur d'un type ASN.1 mais ne l'étant pas nécessairement; et
- b) de l'identification (jointe ou séparée):
 - 1) d'une classe de valeurs contenant cette valeur de donnée (une syntaxe abstraite); et
 - 2) du codage utilisé (la syntaxe de transfert) pour distinguer cette valeur de donnée des autres valeurs de la même classe.

NOTES

1 La valeur de donnée peut être une valeur d'un type ASN.1, ou être par exemple le codage d'une image fixe ou animée. L'identification consiste en un ou deux identificateurs d'objet, ou désigne un contexte de présentation OSI permettant d'identifier la syntaxe abstraite et la syntaxe de transfert.

2 L'identification de la syntaxe abstraite et/ou du codage peut également être assurée par le concepteur de l'application sous la forme d'une valeur fixe, auquel cas elle ne sera pas codée dans une instance de communication.

30.4 Le type pdv encapsulé possède un type associé, qui sert à préciser la définition des valeurs abstraites de ce type, et qui sert également de base à la notation de ses valeurs et sous-types.

NOTE – Les règles de codage peuvent définir un type différent utilisé pour en dériver les codages, ou spécifier les codages sans faire référence à un quelconque type associé. En particulier, les règles de codage en paquet (PER) garantissent que les multiples occurrences (dans un même message) d'un type PDV encapsulé correspondant à un même identificateur d'objet verront cet identificateur d'objet codé une seule fois.

30.5 En supposant un environnement d'étiquetage automatique, le type associé servant à la définition des valeurs et au sous-typage est le suivant (les commentaires ont force de norme):

SEQUENCE {	CHOICE {
identification	SEQUENCE {
syntaxes	OBJECT IDENTIFIER,
abstract	OBJECT IDENTIFIER }
transfer	
-- Identificateurs d'objet de la syntaxe abstraite et de la syntaxe de transfert --,	

30.9 EXEMPLE 1 – Lorsqu'un concepteur d'application souhaite que le codage soit indépendant de tout environnement de présentation (et qu'il soit donc possible de le retransmettre, de le stocker et de l'extraire sans modification), il doit interdire l'utilisation des formes d'identification "presentation-context-id" et "context-negotiation". Il peut y parvenir en écrivant:

```
EMBEDDED PDV (WITH COMPONENTS {
    ... ,
    identification (WITH COMPONENTS {
        ... ,
        presentation-context-id    ABSENT,
        context-negotiation        ABSENT } ) ) )
```

30.10 EXEMPLE 2 – S'il faut forcer le choix d'une seule option, par exemple celui de la forme "syntaxes", on peut y parvenir en écrivant:

```
EMBEDDED PDV (WITH COMPONENTS {
    ... ,
    identification (WITH COMPONENTS {
        syntaxes PRESENT } ) ) )
```

30.11 EXEMPLE 3 – (L'exemple suivant utilise le type paramétré spécifié dans la Rec. UIT-T X.683 | ISO/CEI 8824-4). Un type paramétré ENCRYPTED est défini au moyen d'un type pdv encapsulé de la manière suivante:

```
ENCRYPTED { A-chiffre } ::= EMBEDDED PDV (WITH COMPONENTS {
    identification (WITH COMPONENTS { fixed PRESENT } ),
    -- La valeur "data-value" est n'importe quelle valeur A-chiffre.
    -- La syntaxe de transfert est la syntaxe de transfert de sécurité.
    data-value (WITH COMPONENTS { notation (A-chiffre) } ) ) )
```

Un type, par exemple "CONFIDENTIEL", peut maintenant être chiffré en écrivant "ENCRYPTED {CONFIDENTIEL}". Dans cet exemple, seule la valeur chiffrée serait transmise, sans autre valeur d'identificateur d'objet.

31 Notation du type externe

31.1 Un type externe (voir 3.8.25) est défini par la notation "ExternalType":

```
ExternalType ::= EXTERNAL
```

31.2 Ce type porte l'étiquette de classe universelle numéro 8.

31.3 Ce type regroupe des valeurs constituées:

- a) du codage d'une valeur de donnée simple, pouvant être une valeur d'un type ASN.1 mais ne l'étant pas nécessairement; et
- b) de l'identification:
 - 1) d'une classe de valeurs contenant cette valeur de donnée (une syntaxe abstraite); et
 - 2) du codage utilisé (la syntaxe de transfert) pour distinguer cette valeur de donnée des autres valeurs de la même classe; et
- c) (optionnellement) d'un descripteur d'objet assurant une description en langage naturel de la classe de la valeur de donnée. Le descripteur d'objet optionnel ne sera présent que si le commentaire associé à l'utilisation de la notation "ExternalType" le permet explicitement.

NOTE – La Note 1 du 30.4 s'applique également au type externe.

31.4 Le type externe possède un type associé, qui sert à préciser la définition des valeurs abstraites de ce type, et qui sert également de base à la notation de ses valeurs et sous-types.

NOTE – Les règles de codage peuvent définir un type différent utilisé pour en dériver les codages, ou spécifier les codages sans faire référence à un quelconque type associé. En particulier, les règles de codage de base (BER) utilisent un type séquence équivalent identique à celui qui est présent dans la définition du type externe de la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990, et les codages des valeurs externes par les règles BER restent inchangées.

31.5 En supposant un environnement d'étiquetage automatique, le type associé servant à la définition des valeurs et au sous-typage est le suivant (les commentaires ont force de norme):

```

SEQUENCE {
  identification
    CHOICE {
      syntaxes
        SEQUENCE {
          abstract
            OBJECT IDENTIFIER,
          transfer
            OBJECT IDENTIFIER }
        -- Identificateurs d'objet de la syntaxe abstraite et de la syntaxe de transfert --,

      syntax
        OBJECT IDENTIFIER
        -- Identificateur d'objet unique pour identifier la classe et le codage --,

      presentation-context-id
        INTEGER
        -- (Ne s'applique que dans les environnements OSI)
        -- Le contexte de présentation négocié identifie la classe de la valeur et son codage --,

      context-negotiation
        SEQUENCE {
          presentation-context-id
            INTEGER
          transfer-syntax
            OBJECT IDENTIFIER }
        -- (Ne s'applique que dans les environnements OSI)
        -- Négociation de contexte en cours pour identifier
        -- la classe de la valeur et son codage --,

      transfer-syntax
        OBJECT IDENTIFIER
        -- La classe de la valeur (une spécification indiquant par exemple qu'il s'agit d'une valeur
        -- du type ASN.1) est fixée par le concepteur de l'application (et est donc connue à la fois de
        -- l'expéditeur et du destinataire). Ce cas est prévu avant tout pour prendre en charge le
        -- chiffrement sélectif par champ (ou d'autres transformations de codage) d'un type ASN.1 --,

      fixed
        NULL
        -- La valeur de donnée est une valeur d'un type ASN.1 fixe
        -- (qui est donc connu à la fois de l'expéditeur et du destinataire) -- },

    data-value-descriptor
      ObjectDescriptor OPTIONAL
      -- Il s'agit de l'identification en langage naturel de la classe de la valeur --,

    data-value
      CHOICE {
        notation
          ABSTRACT-SYNTAX.&Type
          -- Cette notation de type, définie dans la Rec. UIT-T X.681 | ISO/CEI 8824-2, a une notation
          -- de valeur qui est n'importe quelle définition de type ASN.1, suivie de deux points ":"
          -- et de la notation de valeur correspondant à ce type. Cette notation a été prévue pour
          -- permettre de spécifier dans une notation conviviale des valeurs de données
          -- appartenant à un type ASN.1. --,

        encoded
          BIT STRING
          -- Cette notation a été prévue pour permettre de spécifier des valeurs de données
          -- n'appartenant pas à un type ASN.1 unique. -- } }

  ( WITH COMPONENTS {
    ... ,
    identification (WITH COMPONENTS {
      ... ,
      syntaxes
        ABSENT,
      transfer-syntax
        ABSENT,
      fixed
        ABSENT } ) )
  }

```

NOTE – Le type externe ne permet pas l'utilisation des formes "syntaxes", "transfer-syntax" et "fixed" de la notation "identification", afin de préserver la compatibilité amont avec le type externe défini dans la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990. Les concepteurs d'applications ayant besoin de faire appel à ces formes devront utiliser le type pdv encapsulé. La définition du type associé donnée ici souligne les points communs existant entre le type externe, le type chaîne de caractères sans restriction et le type pdv encapsulé.

31.6 Les dispositions des 30.6 et 30.7 s'appliquent également au type externe.

31.7 Une valeur de type externe est déclarée par la notation de valeur du type associé défini au 31.5.

ExternalValue ::= SequenceValue -- valeur du type associé défini au 31.5

32 Les types chaînes de caractères

Ces types représentent les chaînes de caractères constituées à partir d'un répertoire de caractère spécifié donné. Il est normal de définir un répertoire de caractères et son codage en les représentant sous la forme de cellules disposées en une ou plusieurs grilles, chaque cellule correspondant à un caractère du répertoire. D'habitude, on affecte également à chaque cellule un symbole graphique et un nom de caractère, bien que dans certains répertoires, des cellules soient laissées vides ou qu'elles aient un nom mais pas de forme imprimable (on peut citer comme exemples de telles cellules ayant un nom mais pas de forme affectée les caractères de contrôle comme EOF de ISO 646 et les caractères d'espacement comme THIN-SPACE and EN-SPACE de ISO/CEI 10646-1).

L'expression **caractère abstrait** désigne la totalité de l'information associée à une cellule d'une grille de répertoire de caractères. Cette information décrit un caractère abstrait distinct même si cette information est vide (pas de symbole graphique ni de nom affecté à la cellule).

La notation de valeurs ASN.1 des types chaînes de caractères possède trois variantes (combinables) spécifiées comme suit:

- a) une représentation imprimée des caractères de la chaîne utilisant les symboles graphiques attribués aux caractères, y compris éventuellement les caractères d'espacement; c'est la notation "cstring";

NOTES

1 Une telle représentation peut être ambiguë lorsqu'un même symbole graphique est attribué à plus d'un caractère du répertoire.

2 Une telle représentation peut être ambiguë lorsque la chaîne comprend des caractères d'espacement et que la notation est imprimée dans une police à chasse variable.

- b) une liste des caractères de la valeur de chaîne sous la forme d'une suite des références de valeurs ASN.1 affectées aux caractères; un ensemble de telles références de valeurs est défini dans le module ASN.1 de l'article 35 pour les répertoires de caractères ISO/CEI 10646-1 et IA5String; cette représentation n'est utilisable pour les autres répertoires de caractères que si l'utilisateur affecte leur affecte des références de valeurs au moyen de la notation décrite au point a) ci-dessus ou au point c) ci-dessous;
- c) une liste des caractères de la valeur de chaîne qui identifie chaque caractère abstrait par la position de sa cellule dans la ou les grilles du répertoire; cette forme n'est utilisable que pour des chaînes "IA5String" (alphabet international n° 5), "UniversalString" (chaîne universelle), et "BMPString" (table multilingue).

33 Notation des types chaîne de caractères

33.1 Un type chaîne de caractères (voir 3.8.11) est désigné au moyen de la notation:

CharacterStringType ::= RestrictedCharacterStringType | UnrestrictedCharacterStringType

"RestrictedCharacterStringType" est la notation du type de chaîne de caractères restreinte définie à l'article 34. "UnrestrictedStringType" est la notation du type de chaîne de caractères non restreinte définie au 37.1.

33.2 Les étiquettes portées par les différents types de chaînes de caractères restreintes sont spécifiées au 34.1. L'étiquette du type de chaîne de caractère non restreinte est spécifiée au 37.2.

33.3 Une valeur de chaîne de caractères est déclarée par la notation suivante:

CharacterStringValue ::= RestrictedCharacterStringValue | UnrestrictedCharacterStringValue

La valeur "RestrictedCharacterStringValue" est définie au 34.7. "UnrestrictedCharacterStringValue" est la notation des valeurs de chaîne de caractères non restreinte; elle est défini au 37.6.

34 Définition des types chaînes de caractères restreintes

Cet article définit les types dont les valeurs sont limitées aux séquences de zéro, un ou plusieurs caractères appartenant à une collection de caractères déterminée. d'un jeu donné. Ce type de chaîne de caractères restreinte est désigné au moyen de la notation "RestrictedCharacterStringType":

**RestrictedCharacterStringType ::= BMPString |
 GeneralString |
 GraphicString |
 IA5String |
 ISO646String |**

NumericString |
PrintableString |
TeletexString |
T61String |
UniversalString |
VideotexString |
VisibleString

Chaque forme possible du type "RestrictedCharacterStringType" est définie en spécifiant:

- l'étiquette qui lui est affectée; et
- un nom (par exemple NumericString) par lequel le type est désigné; et
- les caractères du jeu utilisé pour définir le type, par référence à un tableau énumérant les différentes formes graphiques des caractères, ou par référence à un numéro d'enregistrement du registre international des jeux de caractères codés de l'ISO (voir le *Registre international de l'ISO des jeux de caractères codés à utiliser avec les séquences d'échappement*), ou par référence à ISO/CEI 10646-1.

34.1 Le Tableau 3 donne la liste des noms désignant chacun des types de chaînes de caractères, le numéro de l'étiquette de la classe universelle affectée à ce type, le tableau, article ou numéro d'enregistrement de définition, et, le cas échéant, le numéro de la Note relative à cette entrée du tableau. Lorsqu'un nom de type possède un synonyme, celui-ci est indiqué entre parenthèses.

NOTE – L'étiquette affectée à chaque type de chaînes de caractères l'identifie de manière non ambiguë. Il est à noter toutefois que si la notation ASN.1 est utilisée pour définir de nouveaux types à partir de ce type (notamment en utilisant la déclaration IMPLICIT) il sera peut-être impossible de reconnaître ces types sans connaître la définition de type ASN.1.

Tableau 3 – Liste des types de chaînes de caractères restreintes

Nom pour référencer le type de chaîne	Numéro de classe universelle	Numéro d'enregistrement, numéro du tableau ou article de la Rec. UIT-T X.680 ISO/CEI 8824-1 de définition ^{a)}	Notes
NumericString (chaîne numérique)	18	Tableau 4	(1)
PrintableString (chaîne imprimable)	19	Tableau 5	(1)
TeletexString (T61String) chaîne télételex (ou chaîne T61)	20	6, 87, 102, 103, 106, 107, 126, 144, 150, 153, 156, 164, 165, 168 + SPACE + DELETE	(2)
VideotexString (chaîne vidéotex)	21	1, 13, 72, 73, 87, 89, 102, 108, 126, 128, 129, 144, 150, 153, 164, 165, 168 + SPACE + DELETE	(3)
IA5String (chaîne AI n° 5)	22	1, 6 + SPACE + DELETE	
GraphicString (chaîne graphique)	25	Tous les jeux graphiques + SPACE	
VisibleString (ISO646String) [chaîne visible (chaîne ISO 646)]	26	6 + SPACE	
GeneralString (chaîne générale)	27	Tous les jeux graphiques et caractères + SPACE + DELETE	
UniversalString (chaîne universelle)	28	Voir 34.6	
BMPString (chaîne BMP ou multilingue)	30	Voir 34.12	

^{a)} Les numéros d'enregistrement de définition sont énumérés dans le *Registre international de l'ISO des jeux de caractères codés à utiliser avec les séquences d'échappement*.

NOTES

1 Le style typographique, le corps, la couleur, la graisse et les autres caractéristiques d'impression ou d'affichage ne sont pas significatifs.

2 Les entrées correspondant à ces numéros d'enregistrement renvoient à la Recommandation T.61, pour les règles concernant leur utilisation. Les entrées de registre 6 et 156 peuvent être utilisées à la place des entrées 102 et 103.

3 Les entrées correspondant à ces numéros d'enregistrement assurent les fonctionnalités définies dans les Recommandations T.100 et T.101.

4 La référence au registre 6 du «Registre international de l'ISO des jeux de caractères codés à utiliser avec la séquence d'échappement» est une référence indirecte ISO 646:1991. Il s'agit là d'une modification par rapport à la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990, qui donnait en référence le registre 2 (référence indirecte à ISO 646:1973). S'il est nécessaire de faire référence dans une application au registre n° 2, il faudra recourir à d'autres moyens (utiliser par exemple une chaîne de caractère sans restriction (voir l'article 37) pour véhiculer l'ancienne définition de la chaîne visible ou pour faire référence à la Rec. X.208 du CCITT | ISO/CEI 8824.

34.2 Le Tableau 4 donne la liste des caractères susceptibles d'apparaître dans les chaînes de type "NumericString" (chaînes numériques) et dans leur syntaxe abstraite de caractères.

Tableau 4 – Chaînes numériques "NumericString"

Nom	Caractère graphique
Chiffres	0, 1, ... 9
Espacement	(espacement)

34.3 Les valeurs d'identificateur d'objet et de descripteur d'objet suivantes ont été affectées pour identifier et décrire la syntaxe abstraite de caractères "NumericString" (chaînes numériques):

{ joint-iso-ccitt asn1(1) specification(0) characterStrings(1) numericString(0) }

et

"NumericString character abstract syntax"

NOTES

- 1 Cette valeur d'identificateur d'objet peut être utilisée dans des valeurs de type chaîne de caractère CHARACTER STRING lorsqu'il est nécessaire de transmettre l'identificateur du type de chaîne de caractères indépendamment de la valeur.
- 2 Une valeur de syntaxe abstraite de caractère de chaîne numérique peut être codée:
 - a) par une des règles données dans ISO/CEI 10646-1 pour le codage des caractères abstraits. Dans ce cas, la syntaxe de transfert de caractères est identifiée par l'identificateur d'objet associé aux règles dans l'Annexe M de ISO/CEI 10646-1.
 - b) par les règles de codage ASN.1 du type prédéfini NumericString. Dans ce cas, la syntaxe de transfert de caractères est identifiée par la valeur d'identificateur d'objet {joint-iso-ccitt asn1(1) basic-encoding(1)}.

34.4 Le Tableau 5 donne la liste des caractères susceptibles d'apparaître dans les chaînes de type "PrintableString" (chaînes imprimables) et dans leur syntaxe abstraite de caractères.

Tableau 5 – Chaînes imprimables "PrintableString"

Nom	Symbole
Majuscules	A, B, ... Z
Minuscules	a, b, ... z
Chiffres	0, 1, ... 9
Espacement	(espacement)
Apostrophe	'
Parenthèse gauche	(
Parenthèse droite)
Signe plus	+
Virgule	,
Trait d'union	-
Point	.
Barre oblique	/
Deux-points	:
Signe égal	=
Point d'interrogation	?

34.5 Les valeurs d'identificateur d'objet et de descripteur d'objet suivantes ont été affectées pour identifier et décrire la syntaxe abstraite de caractères "PrintableString" (chaîne imprimable):

```
{ joint-iso-ccitt asn1(1) specification(0) characterStrings(1) printableString(1) }
```

et

"PrintableString character abstract syntax"

NOTES

- 1 Cette valeur d'identificateur d'objet peut être utilisée dans des valeurs de type chaîne de caractère CHARACTER STRING lorsqu'il est nécessaire de transmettre l'identificateur du type de chaîne de caractères indépendamment de la valeur.
- 2 Une valeur de syntaxe abstraite de caractère de chaîne imprimable peut être codée:
 - a) par une des règles données dans ISO/CEI 10646-1 pour le codage des caractères abstraits. Dans ce cas, la syntaxe de transfert de caractères est identifiée par l'identificateur d'objet associé aux règles dans l'Annexe M de ISO/CEI 10646-1.
 - b) par les règles de codage ASN.1 du type prédéfini PrintableString. Dans ce cas, la syntaxe de transfert de caractères est identifiée par la valeur d'identificateur d'objet {joint-iso-ccitt asn1(1) basic-encoding(1)}.

34.6 Les caractères susceptibles d'apparaître dans les chaînes de type "UniversalString" (chaînes universelles) sont tous les caractères autorisés par ISO/CEI 10646-1. L'utilisation de ce type impose le respect des spécifications de conformité indiquées dans cette norme, notamment en ce qui concerne la zone d'utilisation restreinte.

NOTES

- 1 L'utilisation de ce type sans contrainte est déconseillée, car la conformité sera généralement difficile à assurer.
- 2 L'article 35 contient un module ASN.1 définissant un certain nombre de sous-types de ce type pour les "collections de caractères graphiques des sous-jeux" définies à l'Annexe A de ISO/CEI 10646-1.

34.7 Une valeur de type chaîne de caractères restreinte est déclarée par la notation "cstring" (voir 9.11), "CharacterStringList", "Quadruple" ou "Tuple". "Quadruple" ne peut définir qu'une chaîne de caractères de longueur 1, et ne peut être utilisée que pour noter une valeur de type chaîne universelle "UniversalString" ou multilingue "BMPString". "Tuple" ne peut définir qu'une chaîne de caractères de longueur 1, et ne peut être utilisée que pour noter une valeur de type chaîne AI n° 5 "IA5String".

RestrictedCharacterStringValue ::= cstring | CharacterStringList | Quadruple | Tuple

CharacterStringList ::= "{" CharSyms "}"

CharSyms ::= CharsDefn | CharSyms "," CharsDefn

CharsDefn ::= cstring | DefinedValue

Quadruple ::= "{" Group "," Plane "," Row "," Cell "}"

Group ::= number

Plane ::= number

Row ::= number

Cell ::= number

Tuple ::= "{" TableColumn "," TableRow "}"

TableColumn ::= number

TableRow ::= number

NOTES

1 La notation "cstring" ne peut être utilisée que sur un dispositif capable d'afficher les symboles graphiques des caractères figurant dans la valeur. Réciproquement, si le dispositif utilisé n'offre pas une telle capacité, le seul moyen de spécifier une valeur de chaîne de caractères utilisant de tels symboles est la notation de liste "CharacterStringList", sous réserve que la chaîne soit de type universel "UniversalString", international "BMPString" ou alphabet international n°5 "IA5String", et seulement si la forme "DefinedValue" de "CharsDefn" est utilisée (voir 35.1.2).

2 L'article 35 définit un certain nombre de références "valuereference" désignant des caractères simples (chaînes de longueur 1) du type universel "UniversalString" ou AI n° 5 "IA5String".

EXEMPLE – Supposons que l'on souhaite spécifier une valeur de chaîne universelle "UniversalString" "abcΣdef" dans laquelle le caractère "Σ" ne peut pas être représenté sur le dispositif utilisé. Cette valeur peut alors s'écrire comme suit:

```
IMPORTS BasicLatin, greekCapitalLetterSigma FROM ASN1-CHARACTER-MODULE
```

```
{ joint-iso-ccitt asn1(1) specification(0) modules(0) iso10646(0) };
```

```
Alphabet-Personnel ::= UniversalString (FROM (BasicLatin | greekCapitalLetterSigma))
```

```
chaîne-écrite Alphabet-Personnel ::= { "abc" , greekCapitalLetterSigma , "def" }
```

Remplacée par une version plus récente ISO/CEI 8824-1 : 1995 (F)

3 Lorsqu'on spécifie une valeur du type universel "UniversalString", la notation "cstring" ne sera pas utilisée à moins que n'aient été résolues les ambiguïtés résultant de la similitude de forme de caractères différents.

EXEMPLE – La notation suivante en chaîne "cstring" ne doit pas être utilisée car les symboles graphiques 'P', 'O' et 'T' existent dans les alphabets BASIC LATIN, CYRILLIC et BASIC GREEK, et sont donc ambigus.

```
IMPORTS BasicLatin, Cyrillic, BasicGreek FROM ASN1-CHARACTER-MODULE
{ joint-iso-ccitt asn1(1) specification(0) modules(0) iso10646(0) };
```

```
Alphabet-Personnel ::= UniversalString (FROM (BasicLatin | Cyrillic | BasicGreek))
```

```
chaîne-écrite Alphabet-Personnel ::= "POT"
```

34.8 La valeur "DefinedValue" de "CharsDefn" renverra à une valeur de ce type.

34.9 Les numéros "number" dans les productions "Group", "Plane", "Row" et "Cell" seront inférieurs à 256.

34.10 "Group" spécifie un groupe de l'espace de codage du système de codage universel UCS, "Plane" spécifie un plan du groupe, "Row" spécifie une rangée dans le plan, et "Cell" spécifie une cellule dans la rangée. Le caractère abstrait identifié par cette notation est celui de la cellule spécifiée par les valeurs "Group", "Plane", "Row", et "Cell". Dans tous les cas, l'ensemble des caractères autorisés peut être restreint par sous-typage.

NOTE – Les concepteurs d'applications doivent faire attention aux problèmes de conformité qui pourraient se poser lorsqu'ils utilisent des types chaînes de caractères ouverts tels que des chaînes générales "GeneralString" ou graphiques "GraphicString" sans leur appliquer de contraintes. Un texte précis sur la conformité est aussi nécessaire pour les types chaînes de caractères limités mais longs, tels que les chaînes télétext "TeletexString".

34.11 Le numéro "number" dans la production "TableColumn" sera compris entre zéro et sept, et Le numéro "number" dans la production "TableRow" sera compris entre zéro et quinze. "TableColumn" désigne une colonne et "TableRow" une rangée de la grille des codes des caractères conformément à la disposition représentée Figure 1 de ISO/CEI 2022. Cette notation n'est utilisée que pour les chaînes du type "IA5String" lorsque la grille de codage contient l'entrée de registre 1 en colonnes 0 et 1 et l'entrée de registre 6 dans les colonnes 2 à 7 (voir le *Registre international de l'ISO des jeux de caractères codés à utiliser avec les séquences d'échappement*).

34.12 Le type chaîne multilingue "BMPString" est un sous-type du type chaîne universelle "UniversalString" qui possède une étiquette en propre et modélise la table multilingue BMP (*basic multilingual plane*) de la grille ISO/CEI 10646-1 (les premières 64K cellules à 2 octets). Son type associé est:

```
UniversalString (Bmp)
```

où Bmp est défini dans le module ASN1-CHARACTER-MODULE (voir l'article 35) comme le sous-type du type "UniversalString" correspondant au nom de collection "BMP" défini à l'Annexe A de ISO/CEI 10646-1.

NOTES

1 Comme BMPString est un type prédéfini, il n'est pas défini dans le module ASN1-CHARACTER-MODULE.

2 La définition du type BMPString en tant que type prédéfini a pour but de permettre aux règles de codage (comme les règles de base BER) obéissant à des contraintes d'utiliser un codage sur 16 bits plutôt que sur 32 bits.

3 Toutes les valeurs BMPString sont également des notations de valeurs UniversalString valides.

35 Dénomination des caractères et collections de caractères définis dans ISO/CEI 10646-1

Le présent article définit un module ASN.1 prédéfini qui donne la définition d'un nom de référence de valeur pour chaque caractère défini dans ISO/CEI 10646-1. Chacun de ces noms renvoie à une valeur de chaîne universelle "UniversalString" de taille 1. Ce module définit également un nom de référence de type pour chacune des collections de caractères de ISO/CEI 10646-1, chacun de ces noms renvoyant à un sous-ensemble du type "UniversalString".

NOTE – Ces valeurs peuvent être utilisées pour noter les valeurs du type "UniversalString" et des sous-types qui en dérivent. Toutes les références de valeurs et de types spécifiées au 35.1 sont exportées, et doivent être importées par tout module qui les utilise.

35.1 Spécification du module ASN1-CHARACTER-MODULE

Ce module n'est pas reproduit ici en entier. Ce qui a été plutôt été exposé est la manière dont il a été défini.

35.1.1 Le module commence de la façon suivante:

```
ASN1-CHARACTER-MODULE {joint-iso-ccitt asn1(1) specification(0) modules(0) iso10646(0)}
DEFINITIONS ::= BEGIN
-- Toutes les références de valeurs et de types définies dans ce module sont implicitement exportées
-- et peuvent être importées par tout module.
-- caractères de contrôle de ISO 646

nul IA5String ::= {0, 0}
soh IA5String ::= {0, 1}
stx IA5String ::= {0, 2}
etx IA5String ::= {0, 3}
eot IA5String ::= {0, 4}
enq IA5String ::= {0, 5}
ack IA5String ::= {0, 6}
bel IA5String ::= {0, 7}
bs IA5String ::= {0, 8}
ht IA5String ::= {0, 9}
lf IA5String ::= {0,10}
vt IA5String ::= {0,11}
ff IA5String ::= {0,12}
cr IA5String ::= {0,13}
so IA5String ::= {0,14}
si IA5String ::= {0,15}
dle IA5String ::= {1, 0}
dc1 IA5String ::= {1, 1}
dc2 IA5String ::= {1, 2}
dc3 IA5String ::= {1, 3}
dc4 IA5String ::= {1, 4}
nak IA5String ::= {1, 5}
syn IA5String ::= {1, 6}
etb IA5String ::= {1, 7}
can IA5String ::= {1, 8}
em IA5String ::= {1, 9}
sub IA5String ::= {1,10}
esc IA5String ::= {1,11}
is4 IA5String ::= {1,12}
is3 IA5String ::= {1,13}
is2 IA5String ::= {1,14}
is1 IA5String ::= {1,15}
del IA5String ::= {7,15}
```

35.1.2 Pour chaque entrée de chaque liste de noms de caractères graphiques (glyphes) figurant dans les articles 24 et 25 de ISO/CEI 10646-1, le modèle inclut une déclaration de la forme:

```
<namedcharacter> BMPString ::= <tablecell>
-- représentant le caractère <iso10646name>, voir ISO/CEI 10646-1
```

où:

- a) <iso10646name> est le nom du caractère dérivé de la liste donnée dans ISO/CEI 10646-1;
- b) <namedcharacter> est une chaîne obtenue en appliquant les procédures spécifiées au 35.2 au caractère <iso10646name>;
- c) <tablecell> est le glyphe de la cellule du tableau de ISO/CEI 10646-1 correspondant à l'entrée dans la liste.

EXEMPLE:

```
latinCapitalLetterA BMPString ::= {0, 0, 0, 65}
-- représente le caractère A MAJUSCULE LATIN, voir ISO/CEI 10646-1
greekCapitalLetterSigma BMPString ::= {0, 0, 3, 145}
-- représente le caractère SIGMA MAJUSCULE GREC, voir ISO/CEI 10646-1
```

35.1.3 Pour chaque nom de collection de caractères graphiques spécifié dans l'Annexe A de ISO/CEI 10646-1, le module comporte une déclaration de la forme:

```
<namedcollectionstring> ::= BMPString (FROM (<alternativelist>))  
    -- représente la collection de caractères <collectionstring>, -- voir ISO/CEI 10646-1.
```

où:

- a) <collectionstring> est le nom de la collection de caractères affecté par ISO/CEI 10646-1;
- b) <namedcollectionstring> est formé en appliquant à <collectionstring> les procédures du 35.3;
- c) <alternativelist> est formé en utilisant les caractères <namedcharacter> générés comme l'indique le 35.2 pour chacun des caractères spécifiés dans ISO/CEI 10646-1.

La référence de type résultante <namedcollectionstring> forme un sous-ensemble limité de la collection (voir les exemples didactiques de l'Annexe G).

NOTE – Un sous-ensemble limité est une liste de caractères appartenant à un sous-ensemble spécifié, par opposition à un sous-ensemble sélectionné, qui est une des collections de caractères énumérées à l'Annexe A de ISO/CEI 10646-1 plus la collection latine de base.

EXEMPLE (partiel):

```
space BMPString ::= {0, 0, 0, 32}  
exclamationMark BMPString ::= {0, 0, 0, 33}  
quotationMark BMPString ::= {0, 0, 0, 34}  
... -- et ainsi de suite  
tilde BMPString ::= {0, 0, 0, 126}
```

```
BasicLatin ::= BMPString  
    (FROM (space  
        | exclamationMark  
        | quotationMark  
        | ... -- et ainsi de suite  
        | tilde)  
    )
```

-- représente la collection de caractères latins de base, voir ISO/CEI 10646-1.

-- Les points de suspension, utilisés dans cet exemple par souci d'abréviation, signifient "et ainsi de

-- suite"; il ne faut pas les utiliser sous cette forme dans un véritable module ASN.1.

35.1.4 ISO/CEI 10646-1 définit trois niveaux de réalisation. Tous les types définis dans le module ASN1-CHARACTER-MODULE à l'exception de "Level1" et de "Level2" sont par défaut conformes à une réalisation de niveau 3, car de tels types n'imposent aucune restriction quant à l'utilisation des caractères de combinaison. "Level1" indique qu'une réalisation de niveau 1 est requise; "Level2" indique qu'une réalisation de niveau 2 est requise, et "Level3" indique qu'une réalisation de niveau 3 est requise. Ces types sont définis dans le module ASN1-CHARACTER-MODULE:

```
Level1 ::= BMPString (FROM (ALL EXCEPT CombiningCharacters))
```

```
Level2 ::= BMPString (FROM (ALL EXCEPT CombiningCharactersB-2))
```

```
Level3 ::= BMPString
```

NOTES

1 "CombiningCharacters" et "CombiningCharactersB-2" sont les chaînes de collections nommées <namedcollectionstring> correspondant respectivement aux caractères de combinaison (COMBINING CHARACTERS) et aux caractères de combinaison de type 2 (COMBINING CHARACTERSB-2) définies à l'Annexe A de ISO/CEI 10646-1.

2 "Level1" et "Level2" sont utilisés à la suite d'un signe "IntersectionMark" (voir l'article 44) ou comme seule contrainte dans une spécification "ConstraintSpec". Un exemple en est donné au F.2.7.1.

3 Pour plus de détails, voir G.2.5.

35.1.5 Le module se termine par la déclaration:

END

35.1.6 Un équivalent défini par l'utilisateur de l'exemple donné au 35.1.3 pourrait être:

BasicLatin ::= BMPString (FROM (space..tilde))

-- représente la collection de caractères latine de base (BASIC LATIN) voir ISO/CEI 10646-1.

35.2 Un nom <namedcharacter> est une chaîne dérivée d'un nom <iso10646name> (voir 35.1.2) en lui appliquant l'algorithme suivant:

- a) chaque majuscule de <iso10646name> est mise en minuscule, sauf si elle est précédée par un caractère d'espace, auquel cas elle reste inchangée;
- b) les chiffres et les traits d'union restent inchangés;
- c) les caractères d'espace sont supprimés.

NOTE – L'algorithme ci-dessus, appliqué conjointement avec les directives d'appellation des caractères énoncées à l'Annexe K de ISO/CEI 10646-1, produira toujours une notation de valeur non ambiguë pour chacun des noms de caractère figurant dans ISO/CEI 10646-1.

EXEMPLE – Le caractère ISO/CEI 10646-1, rangée 0, cellule 60, appelé "signe inférieur à" (LESS-THAN SIGN), dont la représentation graphique est "<", peut être désigné par la chaîne de type "DefinedValue":

less-thanSign

35.3 Le nom <namedcollectionstring> est la chaîne dérivée du nom <collectionstring> en lui appliquant l'algorithme suivant:

- a) chaque majuscule du nom de la collection de ISO/CEI 10646-1 est mise en minuscule, sauf si elle est précédée par un caractère d'espace ou est la première lettre du nom, auquel cas elle reste inchangée;
- b) les chiffres et les traits d'union restent inchangés;
- c) les caractères d'espace sont supprimés.

EXEMPLES

1 La collection identifiée dans l'Annexe A de ISO/CEI 10646-1 sous le nom:

BASIC LATIN

a la référence de type ASN.1

BasicLatin

2 Un type chaîne de caractères composé des caractères de la collection BASIC LATIN, ainsi que de la collection BASIC ARABIC peut être défini de la manière suivante:

Chaîne-Donnée-De-Caractères ::= BMPString (FROM (BasicLatin | BasicArabic))

NOTE – La formulation ci-dessus est nécessaire car la formulation apparemment plus simple:

Chaîne-Donnée-De-Caractères ::= BMPString (BasicLatin | BasicArabic)

ne permettrait d'écrire que des chaînes dont tous les caractères appartiennent à la collection soit latine BASIC LATIN soit arabe BASIC ARABIC mais pas à un mélange des deux.

36 Ordre canonique des caractères

36.1 A des fins de sous-typage par la notation d'intervalle de valeurs "ValueRange" et d'utilisation possible par les règles de codage, un ordre canonique des caractères a été spécifié pour les types de chaînes universelles UniversalString, multilingues BMPString, numériques NumericString, imprimables PrintableString, visibles VisibleString et AI n° 5 IA5String.

36.2 Aux fins du seul article présent, les caractères sont en correspondance biunivoque avec les cellules d'une grille de codage, qu'un nom ou une forme de caractère ait été ou non affecté à ces cellules, qu'il s'agisse de caractères de contrôle ou de caractères imprimables, et qu'il s'agisse ou non de caractères de combinaison.

36.3 L'ordre canonique d'un caractère abstrait est défini par l'ordre canonique de sa cellule.

36.4 Pour les chaînes universelles UniversalString, l'ordre canonique des cellules est défini par (voir ISO/CEI 10646-1):

$$256*(256*(256*(\text{numéro de groupe})+(\text{numéro de plan}))+(\text{numéro de rangée}))+(\text{numéro de cellule})$$

Le jeu de caractères complet contient exactement $256*256*256*256$ caractères. Les extrémités d'intervalle de valeurs "ValueRange" dans les notations d'alphabet permis "PermittedAlphabet" (ou les différents caractères pris individuellement) peuvent être désignés soit en utilisant la référence de valeur ASN.1 définie dans le module ASN1-CHARACTER-MODULE, soit (lorsque le symbole graphique n'est pas ambigu dans le contexte de la spécification) en indiquant le symbole graphique dans une chaîne "cstring" (Le module ASN1-CHARACTER-MODULE est défini au 35.1). Il n'est pas possible de spécifier une cellule comme extrémité d'intervalle ou de la désigner individuellement lorsque aucun nom ou symbole graphique n'a été affecté à cette cellule.

36.5 Pour les chaînes multilingues BMPString, l'ordre canonique des cellules est défini par (voir ISO/CEI 10646-1):

$$256*(\text{numéro de rangée})+(\text{numéro de cellule})$$

Le jeu de caractères complet contient exactement $256*256$ caractères. Les extrémités d'intervalle de valeurs "ValueRange" dans les notations d'alphabet permis "PermittedAlphabet" (ou les différents caractères pris individuellement) peuvent être désignés soit en utilisant la référence de valeur ASN.1 définie dans le module ASN1-CHARACTER-MODULE, soit (lorsque le symbole graphique n'est pas ambigu dans le contexte de la spécification) en indiquant le symbole graphique dans une chaîne "cstring". Il n'est pas possible de spécifier une cellule comme extrémité d'intervalle ou de la désigner individuellement lorsque aucun nom ou symbole graphique n'a été affecté à cette cellule.

36.6 Pour les chaînes de type numérique NumericString, l'ordre canonique est défini dans un ordre croissant de gauche à droite par (voir le Tableau 4 de l'article 34):

(espacement) 0 1 2 3 4 5 6 7 8 9

Le jeu de caractères complet contient exactement 11 caractères. L'extrémité d'un intervalle de valeur "ValueRange" (ou un caractère isolé) peut être désignée en indiquant le symbole graphique correspondant dans une chaîne "cstring".

NOTE – Cet ordre est le même que les caractères correspondants de la collection latine de base BASIC LATIN de ISO/CEI 10646-1.

36.7 Pour les chaînes de type imprimable PrintableString, l'ordre canonique est défini dans un ordre croissant de gauche à droite puis de haut en bas par (voir le Tableau 5 de l'article 34):

(Espace) (Apostrophe) (Parenthèse gauche) (Parenthèse droite) (Signe plus) (Virgule) (Trait d'union)
(Point) (Barre oblique) 0123456789 (Deux-points) (Signe égal) (Point d'interrogation)
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

Le jeu de caractères complet contient exactement 74 caractères. L'extrémité d'un intervalle de valeur "ValueRange" (ou un caractère isolé) peut être désignée en indiquant le symbole graphique correspondant dans une chaîne "cstring".

NOTE – Cet ordre est le même que les caractères correspondants de la collection latine de base BASIC LATIN de ISO/CEI 10646-1.

36.8 Pour les chaînes de type visible VisibleString, l'ordre canonique des cellules est défini à partir du codage ISO 646 (appelé ISO 646 ENCODING) de la manière suivante:

(ISO 646 ENCODING) - 32

NOTE – C'est-à-dire que l'ordre canonique est celui des caractères correspondants dans les cellules 2/0 à 7/14 de la grille de codage ISO 646.

Le jeu de caractères complet contient exactement 95 caractères. L'extrémité d'un intervalle de valeur "ValueRange" (ou un caractère isolé) peut être désignée en indiquant le symbole graphique correspondant dans une chaîne "cstring".

36.9 Pour les chaînes de type alphabet international n° 5 IA5String, l'ordre canonique des cellules est défini à partir du codage ISO 646 de la manière suivante:

(ISO 646 ENCODING)

Le jeu de caractères complet contient exactement 128 caractères. L'extrémité d'un intervalle de valeur "ValueRange" (ou un caractère isolé) peut être désignée en indiquant le symbole graphique correspondant dans une chaîne "cstring" ou une référence de valeur de caractère de contrôle ISO 646 définie au 35.1.1.

37 Définition des types chaînes de caractères non restreintes

Cet article définit un type dont les valeurs sont celles de n'importe quelle syntaxe abstraite de caractères. Cette syntaxe abstraite peut faire partie de l'ensemble contextuel défini fixé dans une instance de communication, ou être indiquée en référence directement dans chaque instance d'utilisation du type chaîne de caractère non restreinte.

NOTES

1 Une syntaxe abstraite de caractères (et une ou plusieurs syntaxes correspondantes de transfert de caractères) peut être définie par tout organisme habilité à attribuer des identificateurs d'objets ASN.1.

2 Les profils établis par une communauté d'intérêt détermineront normalement les syntaxes abstraites de caractères et les syntaxes de transfert de caractères devant être prises en charge pour des instances ou groupes d'instances donnés de chaînes de caractères CHARACTER STRING. On indiquera généralement les syntaxes admises dans un formulaire PICS (déclaration de conformité d'une instance de protocole). Il est à noter que le groupement des instances aux fins de la spécification de la couche application pourra être réalisé à l'aide de différentes références à des types ASN.1 (chacun étant une référence au type chaîne de caractères CHARACTER STRING).

37.1 Le type chaîne de caractères non restreinte (voir 3.8.58) est désigné au moyen de la notation "UnrestrictedCharacterStringType":

UnrestrictedCharacterStringType ::= CHARACTER STRING

37.2 Ce type porte l'étiquette de classe universelle numéro 29.

37.3 Le type chaîne de caractères non restreinte possède un type associé qui sert uniquement à spécifier la notation de ses valeurs et sous-types.

NOTE – Les règles de codage peuvent définir un type différent utilisé pour en dériver les codages des valeurs de ce type.

37.4 En supposant un environnement d'étiquetage automatique, le type associé servant à la définition des valeurs et au sous-typage est le suivant (les commentaires ont force de norme):

```
SEQUENCE {
  identification
  syntaxes
    abstract
    transfer
    -- Identificateurs d'objet de la syntaxe abstraite et de la syntaxe de transfert --,
  syntax
    -- Identificateur d'objet unique pour identifier la classe et le codage --,
  presentation-context-id
    -- (Ne s'applique que dans les environnements OSI)
    -- Le contexte de présentation négocié identifie la classe de la valeur et son codage --,
  context-negotiation
    presentation-context-id
    transfer-syntax
    -- (Ne s'applique que dans les environnements OSI)
    -- Négociation de contexte en cours pour identifier
    -- la classe de la valeur et son codage --,
  transfer-syntax
    -- La classe de la valeur (une spécification indiquant par exemple qu'il s'agit d'une valeur
    -- du type ASN.1) est fixée par le concepteur de l'application (et est donc connue à la fois de
    -- l'expéditeur et du destinataire). Ce cas est prévu avant tout pour prendre en charge le
    -- chiffrement sélectif par champ (ou d'autres transformations de codage) d'un type ASN.1 --,
  fixed
    -- La valeur de donnée est une valeur d'un type ASN.1 fixe
    -- (qui est donc connu à la fois de l'expéditeur et du destinataire) -- },
  data-value-descriptor
    -- Il s'agit de l'identification en langage naturel de la classe de la valeur --,
```

string-value **CHOICE {**
notation **ABSTRACT-SYNTAX.&Type**
-- Cette notation de type, définie dans la Rec. UIT-T X.681 | ISO/CEI 8824-2, a une notation
-- de valeur qui est n'importe quelle définition de type ASN.1, suivie de deux points ":"
-- et de la notation de valeur correspondant à ce type. Cette notation a été prévue pour
-- permettre de spécifier dans une notation conviviale des valeurs de données
-- appartenant à un type ASN.1. --,
encoded **OCTET STRING**
-- Cette notation a été prévue pour permettre de spécifier des valeurs de données
-- n'appartenant pas à un type ASN.1. -- }
(WITH COMPONENTS {
... ,
data-value-descriptor ABSENT })

NOTE – Le type chaîne de caractères non restreinte ne permet pas d'inclure une valeur de descripteur de valeur de donnée "data-value-descriptor" en même temps qu'une "identification". Toutefois, la définition du type associé donnée ici souligne les points communs existant entre le type pdv encapsulé, le type externe et le type chaîne de caractères non restreinte.

37.5 La valeur "string-value" sera indiquée soit par la forme "notation" (seulement si la syntaxe de caractères abstraite est définie comme un type de chaîne de caractères restreinte ou comme un sous-ensemble de ce type), soit par la forme "encoded". Dans ce dernier cas, la composante "identification" identifie une syntaxe de transfert de caractères pour la syntaxe abstraite de caractères, et la composante "encoded" est une représentation de la chaîne de caractère par une chaîne d'octets utilisant la syntaxe de transfert identifiée par le champ "identification".

37.6 Une valeur de type chaîne de caractères non restreinte est déclarée par la notation de valeur de son type associé:

UnrestrictedCharacterStringValue ::= SequenceValue *-- valeur du type associé défini au 37.4*

37.7 Un exemple de chaîne de caractère non restreinte est donné au F.2.8.

38 Notation des types définis dans les articles 39 à 41

38.1 La notation permettant de faire référence à un type défini dans les articles 39 à 41 est la suivante:

TypeUtile ::= référence-de-type

où "référence-de-type" est l'une des références définies dans les articles 39 à 41 en notation ASN.1.

38.2 L'étiquette de chaque "TypeUtile" est spécifiée dans les articles 39 à 41.

39 Temps généralisé

39.1 Ce type est désigné par le nom:

GeneralizedTime

39.2 Ce type est composé de valeurs représentant:

- a) une date calendaire, telle que celle-ci est définie dans ISO 8601;
- b) une heure, à une des précisions définies dans ISO 8601, à l'exception de la valeur horaire 24 qui n'est pas utilisée;
- c) le facteur de décalage horaire local, tel qu'il est défini dans ISO 8601.

39.3 Ce type est défini en ASN.1 comme suit:

GeneralizedTime ::= [UNIVERSAL 24] IMPLICIT VisibleString

les valeurs de chaîne visible "VisibleString" étant restreintes à l'une des chaînes de caractères suivantes:

- a) une chaîne représentant la date calendaire selon les spécifications de ISO 8601, l'année étant représentée par quatre chiffres, le mois par deux chiffres et le jour par deux chiffres, sans caractères séparateurs, suivie d'une chaîne représentant l'heure selon les spécifications de ISO 8601, sans autre caractère séparateur que la virgule ou le point décimal, (comme prévu dans ISO 8601) et sans lettre finale Z (comme le prévoit ISO 8601);
- b) les caractères de l'alinéa a) ci-dessus, suivis d'une lettre Z majuscule;
- c) les caractères de l'alinéa a) ci-dessus, suivis d'une chaîne représentant un décalage horaire local selon les spécifications de ISO 8601, sans caractère séparateur.

Dans le cas a), l'heure correspond à l'heure locale. Dans le cas b), le temps représente l'heure universelle coordonnée (heure UTC). Dans le cas c), la partie de la chaîne formée comme dans le cas a) correspond à l'heure locale (t_1), le décalage horaire (t_2) permettant de déterminer l'heure UTC de la manière suivante:

$$\text{temps universel coordonné (UTC)} = t_1 - t_2$$

EXEMPLES:

Cas a)

"19851106210627.3"

heure locale 21 heures 6 minutes 27,3 secondes, le 6 novembre 1985.

Cas b)

"19851106210627.3Z"

temps universel coordonné correspondant à l'heure ci-dessus.

Cas c)

"19851106210627.3-0500"

même heure locale que dans le cas a), avec un retard local de 5 heures sur l'heure UTC.

39.4 L'étiquette de ce type sera conforme à la définition donnée au 39.3.

39.5 La notation des valeurs sera identique à celle de la chaîne visible "VisibleString" définie au 39.3.

40 Temps universel

40.1 Ce type sera désigné par le nom:

UTCTime

40.2 Le type est composé de valeurs représentant:

- a) une date calendaire;
- b) une heure, à la précision de la minute ou de la seconde;
- c) et optionnellement, le décalage de l'heure locale par rapport au temps universel coordonné.

40.3 Le type est défini en ASN.1 de la manière suivante:

UTCTime ::= [UNIVERSAL 23] IMPLICIT VisibleString

les valeurs de la chaîne visible "VisibleString" étant limitées aux chaînes de caractères résultant de la juxtaposition des éléments suivants:

- a) six chiffres AAMMJJ, où AA représente les deux chiffres de plus faible poids de l'année grégorienne, MM le mois (avec janvier comme mois 01) et JJ le quantième du mois (de 01 à 31); et
- b) l'une des deux chaînes suivantes:
 - 1) quatre chiffres hhmm où hh représente l'heure (00 à 23) et mm les minutes (00 à 59); ou
 - 2) six chiffres hhmmss où hh et mm sont comme en 1) ci-dessus, et ss représente les secondes (00 à 59); et
- c) l'une des deux chaînes suivantes:
 - 1) le caractère Z; ou
 - 2) le caractère + ou le caractère -, suivi de hhmm, hh représentant les heures et mm les minutes.

Les deux formes de l'alinéa b) ci-dessus permettent de spécifier le temps avec différentes précisions.

Dans la forme c) 1), le temps représenté est le temps universel coordonné (UTC). Dans la forme c) 2), le temps t_1 indiqué par les chaînes a) et b) est l'heure locale; le décalage horaire (t_2) indiqué par la chaîne c) 2) permet de déterminer le temps UTC de la manière suivante:

$$\text{Temps universel coordonné} = t_1 - t_2$$

EXEMPLE 1 – Si le temps local est 7 heures du matin du 2 janvier 1982, et que le temps universel coordonné est midi du 2 janvier 1982, la valeur du temps UTCTime est représenté par l'une des chaînes suivantes:

"8201021200Z", ou
"8201020700-0500".

EXEMPLE 2 – Si le temps local est 7 heures du matin du 2 janvier 2001, et que le temps universel coordonné est midi du 2 janvier 2001, la valeur du temps UTCTime est représenté par l'une des chaînes suivantes:

"0101021200Z", ou
"0101020700-0500".

40.4 L'étiquette de ce type sera conforme à la définition donnée au 40.3.

40.5 La notation des valeurs sera identique à celle de la chaîne visible "VisibleString" définie au 40.3.

41 Type descripteur d'objets

41.1 Ce type est désigné par le nom:

ObjectDescriptor

41.2 Ce type est constitué d'un texte en langage naturel décrivant un objet donné. Ce texte ne constitue pas une identification non ambiguë de l'objet, mais des objets différents ne devraient logiquement pas être décrits par un même texte.

NOTE – Il est recommandé que l'autorité affectant un identificateur OBJECT IDENTIFIER à un objet lui affecte également une valeur de descripteur d'objet "ObjectDescriptor".

41.3 Ce type est défini en notation ASN.1 de la manière suivante:

ObjectDescriptor ::= [UNIVERSAL 7] IMPLICIT GraphicString

La chaîne graphique "GraphicString" contient le texte décrivant l'objet.

41.4 L'étiquette est celle qui est définie au 41.3.

41.5 Une valeur sera déclarée par la notation de valeur de chaîne graphique "GraphicString" définie au 41.3.

42 Types contraints

42.1 La notation du type contraint "ConstrainedType" permet soit d'imposer une contrainte à un type (parent) pour en restreindre l'ensemble des valeurs à un sous-type donné, soit d'imposer, dans un type ensemble ou un type séquence, une contrainte relationnelle entre les valeurs de la composante appartenant au type parent et les valeurs des autres composantes de la valeur d'ensemble ou de séquence. Elle permet aussi d'associer un identificateur d'exception à une contrainte.

ConstrainedType ::=
Type Constraint |
TypeWithConstraint

Dans la première forme, le type parent est "Type", et la contrainte est spécifiée par "Constraint" telle que cette contrainte est définie au 42.5. La seconde forme est définie au 42.4.

42.2 Lorsqu'une contrainte est indiquée à la suite d'une notation de type ensemble-de ou séquence-de, elle s'applique au "Type" intérieur à la notation ensemble-de ou séquence-de, et non au type ensemble-de ou séquence-de lui-même.

NOTE – Dans la notation suivante par exemple, la contrainte de taille "(SIZE(1..64))" s'applique à la chaîne visible "VisibleString" et non au type séquence-de:

Nom-des-pays-membres ::= SEQUENCE OF VisibleString (SIZE(1..64))

42.3 Lorsqu'une contrainte est indiquée à la suite d'une notation de type étiqueté "TaggedType", l'interprétation de la notation globale est la même, que l'on considère le type étiqueté ou le type non étiqueté comme le type parent.

42.4 Par suite de l'interprétation donnée au 42.2, une notation spéciale est prévue pour permettre d'appliquer une contrainte à un type ensemble-de ou séquence-de. Il s'agit de la contrainte "TypeWithConstraint":

```

TypeWithConstraint ::=
    SET Constraint OF Type |
    SET SizeConstraint OF Type |
    SEQUENCE Constraint OF Type |
    SEQUENCE SizeConstraint OF Type
  
```

Le type parent est, dans les deux premières formes, "SET OF Type", et dans les deux dernières, "SEQUENCE OF Type". La contrainte est de type général "Constraint" (voir 42.5) dans les première et troisième formes, et porte sur la taille "SizeConstraint" dans les deuxième et quatrième formes (voir 45.5).

NOTE – Bien que la forme "Constraint" recouvre les possibilités de "SizeConstraint", cette dernière forme, non parenthésée, a été prévue pour des considérations de compatibilité amont avec les dispositions de la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990.

42.5 Une contrainte est spécifiée par la notation "Constraint":

```

Constraint ::= "(" ConstraintSpec ExceptionSpec ")"
ConstraintSpec ::=
    SubtypeConstraint |
    GeneralConstraint
  
```

S'il est nécessaire d'imposer une contrainte d'une forme quelconque autre que la contrainte de sous-type "SubtypeConstraint", on utilisera la forme de contrainte générale "GeneralConstraint" spécifiée au 8.1 de la Rec. UIT-T X.682 | ISO/CEI 8824-3; sinon on utilisera la forme "SubtypeConstraint" spécifiée au 42.6. La spécification d'exception "ExceptionSpec" est définie à l'article 43. Elle n'apparaîtra que lorsqu'une occurrence de référence muette "DummyReference" (voir 8.3 de la Rec. UIT-T X.683 | ISO/CEI 8824-4) ou une contrainte définie par l'utilisateur "UserDefinedConstraint" (voir l'article 9 de la Rec. UIT-T X.682 | ISO/CEI 8824-3) figurera dans la spécification de contrainte "ConstraintSpec".

42.6 La notation "SubtypeConstraint" est la notation d'ordre général "ElementSetSpec" (voir l'article 44):

```

SubtypeConstraint ::= ElementSetSpec
  
```

Dans ce contexte, les éléments sont des valeurs du type parent (le gouvernant de l'ensemble est le type parent). L'ensemble comprendra au moins un élément.

43 Identificateur d'exception

43.1 Dans une spécification ASN.1 complexe, il existe plusieurs situations dans lesquelles il est admis que les décodeurs auront à traiter des éléments incomplètement spécifiés. Ces situations résultent notamment de l'utilisation d'une contrainte définie au moyen d'un paramètre de la syntaxe abstraite (voir l'article 10 de la Rec. UIT-T X.683 | ISO/CEI 8824-4).

43.2 Dans de telles situations, les concepteurs d'applications doivent déterminer les mesures à prendre lorsqu'une contrainte donnée dépendant de l'application est transgressée. L'identificateur d'exception est prévu comme un moyen non ambigu de faire référence à des éléments de spécification ASN.1 indiquant les actions à exécuter dans ces cas. L'identificateur est constitué du caractère "!", suivi d'un type ASN.1 optionnel et d'une valeur de ce type. Si le type optionnel n'est pas mentionné, la valeur est supposée être du type entier INTEGER.

43.3 Si un identificateur d'exception est indiqué, il signifie qu'il existe un texte dans le corps de la norme disant comment traiter la transgression de contrainte associé au signe "!". En l'absence d'un tel identificateur et si une transgression de contrainte a lieu, les concepteurs devront soit identifier le texte décrivant les mesures à prendre, soit décider de ces mesures en fonction de l'application.

44.3 La notation "Elements" est comme suit:

```
Elements ::=
  SubtypeElements |
  ObjectSetElements |
  "(" ElementSetSpec ")"
```

Les éléments spécifiés par cette notation sont:

- comme décrit dans l'article 45 ci-dessous si l'expression utilise la forme "SubtypeElements". Cette notation ne sera utilisée que lorsque le gouvernant est un type, et que le type concerné doit servir à imposer des contraintes supplémentaires aux possibilités de notation. Dans un tel contexte, le gouvernant est considéré comme le type parent;
- ceux décrits au 12.3 de la Rec. UIT-T X.681 | ISO/CEI 8824-2 si l'expression utilise la forme "ObjectSetElements". Cette notation ne sera utilisée que lorsque le gouvernant est une classe d'objets informationnels;
- les éléments spécifiés par la notation "ElementSetSpec" si la troisième forme est utilisée.

45 Éléments de sous-type

45.1 Généralités

Plusieurs formes sont prévues pour noter les éléments de sous-type "SubtypeElements". Elles sont identifiées ci-dessous, leur syntaxe et leur sémantique étant définies dans les points suivants. Le Tableau 6 récapitule pour chaque type parent les notations qui peuvent lui être appliquées.

```
SubtypeElements ::=
  SingleValue | -- valeur unique
  ContainedSubtype | -- sous-type contenu
  ValueRange | -- intervalle
  PermittedAlphabet | -- alphabet permis
  SizeConstraint | -- contrainte de taille
  TypeConstraint | -- type constraint
  InnerTypeConstraints -- sous-typage interne
```

45.2 Valeur unique

45.2.1 La notation de valeur unique "SingleValue" est la suivante:

```
SingleValue ::= Value
```

où "Value" est la notation de la valeur appartenant au type parent.

45.2.2 La notation "SingleValue" détermine la valeur unique du type parent spécifiée par "Value".

45.3 Sous-type contenu

45.3.1 La notation de sous-type contenu "ContainedSubtype" est la suivante:

```
ContainedSubtype ::= Includes Type
Includes ::= INCLUDES | empty
```

La forme vide "empty" de la production "Includes" ne sera pas utilisée si le "Type" du sous-type contenu "ContainedSubtype" est la notation du type néant.

45.3.2 La notation "ContainedSubtype" spécifie toutes les valeurs du type parent qui résultent de l'intersection du type parent et de "Type", qui doit lui-même dériver du même type prédéfini que le type parent.

45.4 Intervalle de valeurs

45.4.1 La notation d'intervalle de valeurs "ValueRange" est la suivante:

```
ValueRange ::= LowerEndpoint ".." UpperEndpoint
```

Tableau 6 – Applicabilité de la notation de sous-typage selon le type parent

Type	Valeur unique	Sous-type contenu	Intervalle	Contrainte de taille	Alphabet permis	Type contraint	Sous-typage interne
Chaîne binaire	Oui	Oui	Non	Oui	Non	Non	Non
Booléen	Oui	Oui	Non	Non	Non	Non	Non
Choix	Oui	Oui	Non	Non	Non	Non	Oui
pdv encapsulé	Oui	Non	Non	Non	Non	Non	Oui
Enuméré	Oui	Oui	Non	Non	Non	Non	Non
Externe	Oui	Non	Non	Non	Non	Non	Oui
Instance-de	Oui	Oui	Non	Non	Non	Non	Oui
Entier	Oui	Oui	Oui	Non	Non	Non	Non
Néant	Oui	Oui	Non	Non	Non	Non	Non
Type champ de classe d'objets	Oui	Oui	Non	Non	Non	Non	Non
Identificateur d'objet	Oui	Oui	Non	Non	Non	Non	Non
Chaîne d'octets	Oui	Oui	Non	Oui	Non	Non	Non
Type ouvert	Non	Non	Non	Non	Non	Oui	Non
Réel	Oui	Oui	Oui	Non	Non	Non	Oui
Types de chaînes de caractères restreintes	Oui	Oui	Oui ^{a)}	Oui	Oui	Non	Non
Séquence	Oui	Oui	Non	Non	Non	Non	Oui
Séquence-de	Oui	Oui	Non	Oui	Non	Non	Oui
Ensemble	Oui	Oui	Non	Non	Non	Non	Oui
Ensemble-de	Oui	Oui	Non	Oui	Non	Non	Oui
Types de chaînes de caractères non restreintes	Oui	Non	Non	Oui	Non	Non	Oui

^{a)} Autorisé seulement avec l'alphabet autorisé "PermittedAlphabet" des types chaînes multilingues BMPString, AI n° 5 IA5String, numériques NumericString, imprimables PrintableString, visibles VisibleString et universelles UniversalString.

45.4.2 La notation "ValueRange" spécifie toutes les valeurs d'un intervalle désigné en spécifiant les valeurs de ses extrémités. Cette notation ne peut être appliquée utilisée qu'avec les types d'entiers, de réels et l'alphabet permis "PermittedAlphabet" de certains types de chaînes de caractères restreintes (AI n° 5 IA5String, numériques NumericString, imprimables PrintableString, visibles VisibleString, multilingues BMPString et universelles UniversalString seulement).

NOTE – Pour les besoins du sous-typage, "PLUS-INFINITY" est supérieur et "MINUS-INFINITY" inférieur à toutes les valeurs réelles.

45.4.3 Chaque extrémité d'intervalle est soit bornée (auquel cas elle est spécifiée), soit non bornée (auquel cas elle n'est pas spécifiée). Quand l'extrémité est non bornée, la spécification inclut un symbole inférieur-à ("<"):

LowerEndpoint ::= LowerEndValue | LowerEndValue "<"

UpperEndpoint ::= UpperEndValue | "<" UpperEndValue

45.4.4 Une extrémité peut également ne pas être spécifiée, auquel cas l'intervalle s'étend dans ce sens aussi loin que le type parent l'autorise:

LowerEndValue ::= Value | MIN

UpperEndValue ::= Value | MAX

45.5 Contrainte de taille

45.5.1 La notation de contrainte de taille "SizeConstraint" est la suivante:

SizeConstraint ::= SIZE Constraint

45.5.2 Une contrainte de taille "SizeConstraint" ne peut s'appliquer qu'aux types chaîne binaire, chaîne d'octets, chaîne de caractères, ensemble-de ou séquence-de, ou aux types obtenus à partir de ces derniers par étiquetage.

45.5.3 La contrainte "Constraint" spécifie les valeurs entières autorisées pour la longueur des valeurs spécifiées; elle a la forme de n'importe quelle contrainte pouvant être appliquée au type parent suivant:

INTEGER (0 .. MAX)

La contrainte "Constraint" sera exprimée au moyen de la forme d'une contrainte de sous-type "SubtypeConstraint" dans la notation de spécification de contrainte "ConstraintSpec".

45.5.4 L'unité de mesure dépend du type parent de la manière suivante:

<i>Type</i>	<i>Unité de mesure</i>
Chaîne binaire	bit
Chaîne d'octets	octet
Chaîne de caractères	caractère
Ensemble-de	valeur composante
Séquence-de	valeur composante

NOTE – On distinguera clairement le décompte des caractères utilisé dans cet article pour déterminer la taille des chaînes de caractères, d'un quelconque décompte d'octets. Le décompte des caractères sera interprété selon la définition de la collection de caractères utilisée dans le type, et plus particulièrement par rapport aux normes, grilles ou numéros d'enregistrement dans un registre pouvant apparaître dans une telle définition.

45.6 Contrainte de type

45.6.1 La notation de contrainte de type "TypeConstraint" est la suivante:

TypeConstraint ::= Type

45.6.2 Cette notation ne s'applique qu'à la notation d'un type ouvert et restreint ce type aux valeurs du type "Type".

45.7 Alphabet autorisé

45.7.1 La notation d'alphabet autorisé "PermittedAlphabet" est la suivante:

PermittedAlphabet ::= FROM Constraint

45.7.2 La notation "PermittedAlphabet" spécifie toutes les valeurs qui peuvent être obtenues en utilisant un sous-alphabet de la chaîne parentale. Cette notation ne s'applique qu'aux types chaîne de caractères restreintes.

45.7.3 La contrainte "Constraint" est n'importe quelle contrainte pouvant s'appliquer au type parent (voir le Tableau 6), sauf qu'elle doit utiliser la forme de contrainte de sous-type "SubtypeConstraint" de la production "ConstraintSpec". Le sous-alphabet inclut tous les caractères qui figurent dans une ou plusieurs valeurs du type de chaîne parentale et qui sont autorisés par la contrainte "Constraint" et seulement ces caractères.

45.8 Sous-typage interne

45.8.1 La notation de contrainte de type interne "InnerTypeConstraints" est la suivante:

InnerTypeConstraints ::=
WITH COMPONENT SingleTypeConstraint |
WITH COMPONENTS MultipleTypeConstraints

45.8.2 La notation "InnerTypeConstraints" spécifie les valeurs qui satisfont à un ensemble de contraintes portant sur la présence et/ou la valeur des composantes du type parental. Une valeur du type parental n'est spécifiée que si elle satisfait à toutes les contraintes explicites ou implicites (voir 45.8.6). Cette notation peut être appliquée aux types ensemble-de, séquence-de, ensemble, séquence et choix, ainsi qu'aux types qui en dérivent par étiquetage.

45.8.3 Une contrainte ayant la forme d'une spécification de valeur de sous-type est prévue pour les types (ensemble-de, séquence-de) qui sont définis en termes d'un seul autre type (interne). La notation de cette contrainte de type unique "SingleTypeConstraint" est la suivante.

SingleTypeConstraint ::= Constraint

La contrainte "Constraint" définit un sous-type du type unique (interne). Une valeur du type parental est spécifiée si et seulement si chaque valeur interne appartient au sous-type obtenu en appliquant la contrainte au type interne.

45.8.4 Lorsqu'un type (choix, ensemble, séquence) est défini en termes de plusieurs autres types (internes), plusieurs contraintes peuvent s'appliquer aux types internes qui le composent. La notation de ces contraintes de types multiples "MultipleTypeConstraints" est la suivante:

MultipleTypeConstraints ::= FullSpecification | PartialSpecification

FullSpecification ::= "{" TypeConstraints "}"

PartialSpecification ::= "{" "... " "," TypeConstraints "}"

TypeConstraints ::=

NamedConstraint |

NamedConstraint "," TypeConstraints

NamedConstraint ::=

identifiant ComponentConstraint

45.8.5 Les contraintes de types "TypeConstraints" contiennent une liste de contraintes s'appliquant aux types composants du type parental. Pour un type séquence, les contraintes doivent figurer dans l'ordre de la séquence. Le type interne auquel les contraintes s'appliquent est identifié par son identificateur. Il y aura au plus une contrainte nommée "NamedConstraint" par composante.

45.8.6 Les contraintes de types multiples "MultipleTypeConstraints" comprennent soit une spécification complète "FullSpecification", soit une spécification partielle "PartialSpecification". Lorsqu'une spécification complète "FullSpecification" est donnée, il existe une contrainte de présence implicite selon laquelle la déclaration "ABSENT" est apposée à tous les types internes qui peuvent recevoir une telle contrainte (voir 45.8.9) et qui ne sont pas explicitement énumérés. Lorsqu'une spécification partielle "PartialSpecification" est donnée il n'y a pas de contrainte implicite, et tout type interne peut être omis de la liste.

45.8.7 Les contraintes s'appliquant à un type interne donné peuvent concerner sa présence (exprimée en valeurs du type parental), ses valeurs, ou les deux. La notation de la contrainte de composante "ComponentConstraint" est la suivante:

ComponentConstraint ::= ValueConstraint PresenceConstraint

45.8.8 Une contrainte portant sur la valeur d'un type interne est exprimée par la notation "ValueConstraint":

ValueConstraint ::= Constraint | empty

La contrainte est satisfaite par une valeur du type parental si et seulement si la valeur interne appartient au sous-type spécifié par la contrainte "Constraint" appliquée au type interne.

45.8.9 Une contrainte portant sur la présence d'un type interne est exprimée par la notation "PresenceConstraint":

PresenceConstraint ::= PRESENT | ABSENT | OPTIONAL | empty

La signification de ces formes et les situations dans lesquelles elles sont autorisées sont définies aux 45.8.9.1 à 45.8.9.3.

45.8.9.1 Si le type parent est une séquence ou un ensemble, un type composant déclaré "OPTIONAL" peut être contraint par la déclaration "PRESENT", (auquel cas la contrainte est satisfaite si et seulement si la valeur de composante correspondante figure), par la déclaration "ABSENT" (auquel cas la contrainte est satisfaite si et seulement si la valeur de composante correspondante ne figure pas) ou par la déclaration "OPTIONAL" (auquel cas aucune contrainte n'est imposée quant à la présence de la valeur de composante correspondante).

45.8.9.2 Si le type parent est un choix, un type composant peut être contraint par la déclaration "ABSENT" (auquel cas la contrainte est satisfaite si et seulement si le type composant correspondant n'est pas utilisé dans la valeur), ou par la déclaration "PRESENT" (auquel cas la contrainte est satisfaite si et seulement si le type composant correspondant figure dans la valeur); il y aura au plus un mot-clé "PRESENT" dans une notation de contraintes de types multiples "MultipleTypeConstraints".

NOTE – Le F.4.5 fournit un exemple clarifiant ce passage.

45.8.9.3 La signification d'une contrainte de présence "PresenceConstraint" vide dépend de l'utilisation d'un spécification complète ou partielle:

- a) dans une spécification complète "FullSpecification", elle est équivalente à la contrainte de présence "PRESENT" pour une composante d'ensemble ou de séquence déclarée "OPTIONAL" et n'impose autrement aucune autre contrainte;
- b) dans une spécification partielle "PartialSpecification", aucune contrainte n'est imposée.

Annexe A

Utilisation de la notation ASN.1-88/90

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

A.1 Maintenance

Le terme de **notation ASN.1-88/90** désigne la notation ASN.1 spécifiée par la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990. Le terme de **notation ASN.1 actuelle** désigne la notation spécifiée par la présente Recommandation | Norme internationale.

Au moment de la publication de la présente Recommandation | Norme internationale, la maintenance de la Rec. X.208 du CCITT | ISO/CEI 8824 était encore assurée. Cette maintenance, dont l'existence dépend d'une résolution annuelle du JTC1/SC21 de l'ISO/CEI, ne pourra pas se poursuivre indéfiniment. Elle est assurée pour donner aux utilisateurs de l'ASN.1 le temps de remplacer certains éléments (et en particulier la déclaration ANY et la notation en macro-notations) de la notation ASN.1-88/90 par la notation ASN.1 actuelle (un tel remplacement peut être réalisé sans modification des messages binaires transmis).

A.2 Panachage de l'ASN.1-88/90 et de la notation ASN.1 actuelle

L'ASN.1-88/90 et la notation ASN.1 actuelle spécifient toutes deux une structure syntaxique de haut niveau qui est le module ASN.1. L'utilisateur de l'ASN.1 écrit une série de modules ASN.1 et peut importer des définitions d'autres modules ASN.1.

Dans chaque module, la notation utilisée doit se conformer entièrement soit à la notation ASN.1-88/90 soit à la notation ASN.1 actuelle; une spécification d'utilisateur indiquera clairement la notation adoptée (en se référant à la Recommandation | Norme internationale appropriée) au niveau de chaque module dont le texte est donné dans la Spécification.

A noter qu'il peut arriver qu'un utilisateur souhaite modifier une partie d'un module et d'utiliser à cette fin la nouvelle notation, tout en laissant les autres parties du module dans l'ancienne notation. Ceci est parfaitement possible mais ne peut être réalisé qu'en éclatant le module en deux.

Lorsqu'un module est conforme à la notation ASN.1-88/90, il peut importer des références de types et de valeurs depuis un module écrit en notation ASN.1 actuelle. Ces types et valeurs doivent être associés à des types pouvant être définis au moyen de la seule notation ASN.1-88/90. Par exemple, un module écrit en notation ASN.1-88/90 ne peut importer une valeur du type UniversalString, car celui-ci est défini dans la notation actuelle et non dans la notation ASN.1-88/90; en revanche, il peut importer des valeurs de type INTEGER ou IA5String par exemple.

Lorsqu'un module est conforme à la notation ASN.1 actuelle, il peut importer des références de types et de valeurs depuis un module écrit en notation ASN.1-88/90. Par contre, il n'importera pas de macro-notations ASN.1. La notation de valeur d'un type importé ne sera utilisée dans le module importateur que si les identificateurs des valeurs de type ensemble, séquence et choix utilisées dans la notation des valeurs sont présents, et qu'il n'existe pas dans cette notation de valeur une valeur déclarée avec le type ANY. Une contrainte de type intérieur ne sera pas appliquée à un type importé si la composante sur laquelle porte la contrainte n'a pas d'identificateur.

A.3 Migration vers la notation ASN.1 actuelle

Lorsqu'un module (rédigé à l'origine conformément à la notation ASN.1-88/90) est modifié pour le rendre conforme à la notation ASN.1 actuelle, il faut observer les points suivants:

- a) Chacune des composantes d'un type ensemble, séquence ou choix recevra un identificateur non ambigu dans le cadre de ce type, et ces identificateurs devront être incorporés dans la notation de valeur.

NOTE – La notation de valeur d'un type choix contient le signe deux points (":").

- b) Tous les cas d'utilisation des déclarations ANY et ANY DEFINED BY seront repris avec une définition de classe d'objets informationnels appropriée, les déclarations ANY et ANY DEFINED BY (ainsi que la composante concernée) étant remplacées par des références appropriées à des champs de cette classe d'objets. Dans la plupart des cas, la spécification peut être grandement améliorée par l'insertion judicieuse de contraintes tabulaires et de contraintes relationnelles de composantes. Dans beaucoup de cas, la spécification peut être encore améliorée si les contraintes tabulaires ou les contraintes relationnelles de composantes sont données sous forme de paramètres du type.
- c) Toutes les définitions de macro-notations seront remplacées par la définition d'une classe d'objets informationnels, d'un type paramétré ou d'une valeur paramétrée. Si la déclaration WITH SYNTAX est judicieusement utilisée dans la définition d'une classe d'objets informationnels, la notation utilisée pour définir un objet de cette classe peut être rendue très semblable aux anciennes macro-notations.
- d) Toutes les instances d'utilisation de macro-notations seront remplacées soit par des définitions équivalentes d'objets informationnels, soit par des renvois à des types de champs de classe d'objets, des types paramétrés ou des valeurs paramétrées. Dans la plupart des cas, la spécification d'objets informationnels peut être grandement améliorée en regroupant ces définitions dans des ensembles d'objets informationnels, et en donnant des directives claires quant à l'obligation de prendre en charge tous les objets informationnels de l'ensemble, et à la possibilité pour les applications réceptrices d'apporter des extensions dépendant de l'application à cet ensemble d'objets informationnels, et, dans l'affirmative, la manière de traiter la réception de valeurs "inconnues". Il peut également être souhaitable d'envisager la possibilité d'élargir la classe d'objets informationnels dans une version ultérieure de la Spécification d'utilisateur, et de donner en conséquence des directives aux utilisateurs sur la manière de traiter de telles extensions.
- e) Toutes les occurrences de la déclaration EXTERNAL doivent être examinées avec soin; bien qu'une telle notation reste licite en notation ASN.1 actuelle, une Spécification d'utilisateur peut sans doute être améliorée en procédant de la manière suivante:
- 1) Envisager l'utilisation d'une notation INSTANCE OF (de préférence avec une contrainte tabulaire pouvant prendre la forme d'un paramètre du type, d'une manière similaire à la solution proposée ci-dessus pour les déclarations ANY et ANY DEFINED BY) à la place de la notation EXTERNAL; dans de nombreux cas, cela ne modifiera pas le contenu des messages binaires transmis.
 - 2) Si la notation EXTERNAL est conservée, le recours au sous-typage intérieur du type associé (voir 31.5) peut contribuer à préciser la spécification quant à la possibilité ou non d'utiliser des identificateurs de contexte de présentation. Dans cette situation s'appliquent également les directives données précédemment (voir l'article 31) à propos des valeurs de type EXTERNAL devant être prises en charge, et du traitement par les applications de valeurs non prises en charge à la réception de celles-ci.
 - 3) Envisager le passage à la notation


```
CHOICE {external EXTERNAL, embedded-pdv EMBEDDED PDV }
```

(avec encore une fois un sous-typage interne s'il y a lieu) pour permettre une migration progressive des applications homologues réparties vers la nouvelle notation. Une telle transformation peut affecter les messages binaires transmis, et devrait donc normalement s'effectuer dans le cadre d'un changement de version de protocole. L'utilisation de la déclaration de données encapsulée EMBEDDED PDV (en particulier pour les nouvelles spécifications) devrait normalement conférer plus de souplesse, comme le laisse voir la comparaison des types associés; de plus, le type encapsulé EMBEDDED PDV est codé plus efficacement que le type EXTERNAL par toutes les règles de codage spécifiées dans la Rec. UIT-T X.690 | ISO/CEI 8825-1.
- f) La lisibilité de la notation des modules ASN.1 existants pourra éventuellement être améliorée (sans modification du train binaire transmis) en insérant la déclaration d'étiquetage automatique AUTOMATIC TAGS dans l'en-tête du module et en supprimant tout ou partie des étiquettes déclarées.
- NOTE – Une telle opération doit être effectuée avec prudence, et en ayant à l'esprit la manière dont l'étiquetage automatique fonctionne; incorrectement appliqué, l'étiquetage automatique peut provoquer la modification du train binaire transmis.
- g) Si l'étiquetage automatique n'est pas appliqué à un module existant comme le décrit l'alinéa f) ci-dessus, il sera normalement préférable de ne pas ajouter de nouvelles définitions de types à ce module, mais de créer plutôt un nouveau module (avec étiquetage automatique) pour les nouvelles définitions de types. Ceci permettra de profiter des facilités de l'étiquetage automatique sans affecter le train binaire transmis.

- h) Une attention particulière sera prêté aux champs contenant des chaînes de caractères pour savoir s'il convient d'utiliser la notation CHARACTER STRING, BMPString ou UniversalString. Toutefois, ceci devrait normalement modifier le train binaire transmis, et devrait donc être effectué dans le cadre d'un changement de version.
- i) Les identificateurs "mantissa", "base" et "exponent" doivent être ajoutés à toute notation de valeur réelle utilisant la forme "NumericRealValue" de la production "RealValue". On envisagera de restreindre la base aux valeurs 2 et 10 dans cette notation de type.

D'une manière générale, l'utilisation de la nouvelle notation ASN.1 devrait apporter des améliorations significatives en lisibilité, efficacité, précision et souplesse, notamment si les facilités offertes par les contraintes tabulaires et les contraintes relationnelles de composantes, la paramétrisation et les nouveaux types de chaînes de caractères sont pleinement exploitées. Tous les utilisateurs de la notation ASN.1-88/90 sont expressément invités à procéder à la migration chaque fois qu'une Spécification est révisée, ou, si aucune révision n'est envisagée à terme, dans le cadre d'une activité indépendante.

L'amendement de modules existants en utilisant une notation non conforme à la présente Spécification ASN.1 peut être considéré comme erroné, même s'il est fait référence aux Spécifications ASN.1-88/90 dans ces modules. On évitera en particulier l'utilisation de macro-notations, de déclarations ANY ou ANY DEFINED BY, ainsi que la définition de nouvelles structures d'ensemble, de séquence ou de choix si ces structures ne sont pas pourvues d'identificateurs non ambigus.

Annexe B

Affectation par l'ISO de valeurs de composantes d'identificateur d'objet

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

B.1 Trois arcs dérivent de la racine. L'affectation des valeurs et des identificateurs, ainsi que l'organisme habilité à affecter les valeurs de composantes qui en dérivent sont stipulés comme suit:

<i>Valeur</i>	<i>Identificateur</i>	<i>Organisme habilité à affecter les valeurs des composantes dérivées</i>
0	itu-t	UIT-T
1	iso	ISO
2	joint-iso-itu-t	voir Annexe D

NOTE – La suite de cette annexe ne concerne que l'affectation de valeurs par l'ISO.

B.2 Les identificateurs "itu-t", "iso" et "joint-iso-itu-t" affectés ci-dessus, peuvent chacun être utilisés comme une forme nominative "NameForm".

B.3 Les identificateurs "itu-t" et "joint-iso-itu-t" sont respectivement synonymes des identificateurs "ccitt" et "joint-iso-ccitt", et peuvent donc apparaître dans des valeurs d'identificateurs d'objets.

B.4 Trois arcs dérivent du nœud identifié par "iso". L'affectation des valeurs et des identificateurs est stipulée comme suit:

<i>Valeur</i>	<i>Identificateur</i>	<i>Organisme habilité à affecter les valeurs des composantes dérivées</i>
0	standard	voir B.5
2	member-body	voir B.6
3	identified-organization	voir B.7

Ces identificateurs peuvent être utilisés comme une forme nominative "NameForm".

NOTE – L'arc 1 (registration-authority) n'est pas utilisé. Il avait été réservé dans des versions antérieures à un usage futur, et il n'a pas été retiré par la suite.

B.5 Les arcs dérivés du nœud "standard" portent chacun la valeur du numéro d'une Norme internationale. Si la Norme internationale comporte plusieurs parties, un arc additionnel correspondra au numéro de partie, sauf si une telle disposition est spécifiquement exclue dans le texte de la Norme internationale. Les autres arcs porteront des valeurs selon les stipulations de cette Norme internationale.

NOTE – Si une Norme internationale en une seule partie affecte des identificateurs d'objets, puis est réorganisée ultérieurement en plusieurs parties, les affectations d'identificateurs d'objets continueront de se faire comme si la Norme ne comportait qu'une seule partie.

B.6 Les arcs dérivés du nœud "member-body" porteront comme valeur le code numérique de pays à trois chiffres tel que ce code est spécifié dans ISO 3166, qui identifiera l'organisme membre de l'ISO dans ce pays (voir la Note). Il n'est pas permis d'utiliser la forme nominative "NameForm" de la composante d'identificateur d'objet avec ces identificateurs. La présente Recommandation | Norme internationale ne définit pas d'arc dérivé du code de pays.

NOTE – L'existence d'un code de pays dans ISO 3166 n'implique pas nécessairement qu'il existe un organisme membre de l'ISO représentant ce pays, ou que l'organisme membre de l'ISO représentant ce pays gère un schéma d'affectation de composantes d'identificateurs d'objets.

B.7 Les arcs dérivés du nœud "identified-organization" porteront comme valeur le code de désignateur de code international (ICD) affecté par l'autorité d'enregistrement de ISO 6523, qui identifie une organisation prescriptrice spécifiquement enregistrée par cette autorité comme habilitée à affecter des composantes d'identificateur d'objet (voir Notes 1 et 2). Les arcs dérivant immédiatement d'un nœud ICD porteront une valeur de code d'organisation affectée par l'organisation prescriptrice, conformément à ISO 6523. Les arcs dérivant d'un nœud de code organisation ne sont pas définis par la présente Recommandation | Norme internationale (voir Note 3).

NOTES

1 Le fait de demander que les organisations prescriptrices soient enregistrées auprès de l'autorité d'enregistrement de ISO 6523 en tant qu'entité affectant des codes d'organisation pour des composantes d'identificateur d'objet garantit que seules sont affectées des valeurs numériques conformes à la présente Recommandation | Norme internationale.

2 Le fait de déclarer qu'une organisation prescriptrice affecte des codes d'organisation à des composantes d'identificateurs d'objets n'interdit pas d'utiliser ces codes à d'autres fins.

3 Il est supposé que les organisations identifiées par un code d'organisation définiront d'autres arcs d'une façon qui garantira l'unicité des valeurs affectées.

Annexe C

Affectation par l'UIT-T de valeurs de composantes d'identificateur d'objet

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

C.1 Trois arcs dérivent de la racine. L'affectation des valeurs et des identificateurs, ainsi que l'organisme habilité à affecter les valeurs de composantes qui en dérivent sont stipulés comme suit:

<i>Valeur</i>	<i>Identificateur</i>	<i>Organisme habilité à affecter les valeurs des composantes dérivées</i>
0	itu-t	UIT-T
1	iso	ISO
2	joint-iso-itu-t	voir Annexe D

NOTE – La suite de cette annexe ne concerne que l'affectation de valeurs par l'UIT-T.

C.2 Les identificateurs "itu-t", "iso" et "joint-iso-itu-t" affectés ci-dessus, peuvent chacun être utilisés comme une forme nominative "NameForm".

C.3 Les identificateurs "itu-t" et "joint-iso-itu-t" sont respectivement synonymes des identificateurs "ccitt" et "joint-iso-ccitt", et peuvent donc apparaître dans des valeurs d'identificateurs d'objets.

C.4 Cinq arcs dérivent du nœud identifié par "itu-t". L'affectation des valeurs et des identificateurs est stipulée comme suit:

<i>Valeur</i>	<i>Identificateur</i>	<i>Organisme habilité à affecter les valeurs des composantes dérivées</i>
0	recommandation	voir C.5
1	question	voir C.6
2	administration	voir C.7
3	network-operator	voir C.8
4	identified-organization	voir C.9

Ces identificateurs peuvent être utilisés comme une forme nominative "NameForm".

C.5 Les arcs dérivés du nœud "recommandation" portent les valeurs de 1 à 26, l'identificateur affecté étant une lettre de "a" à "z". Les arcs issus de chacun de ceux-ci portent les numéros des Recommandations UIT-T et CCITT de la série identifiée par cette lettre. Les arcs qui en dérivent ensuite sont définis selon les besoins par les Recommandations concernées. Les identificateurs a à z peuvent être utilisés comme forme nominative "NameForm".

C.6 Les arcs dérivés du nœud "question" portent des valeurs correspondant aux Commissions d'études de l'UIT-T, qualifiées par la période d'études. La valeur est donnée par la formule:

$$\text{numéro de la Commission d'études} + (\text{période} * 32)$$

où "période" a la valeur 0 pour 1984-1988, 1 pour 1988-1992 et ainsi de suite, le multiplicateur étant 32 (en base 10).

Les arcs issus de chaque Commission d'études portent les valeurs correspondant aux numéros des questions inscrites à leur programme de travail, les arcs du niveau suivant étant définis selon les besoins par le groupe auquel est confiée l'étude de la question (groupe de travail ou groupe de rapporteur spécial par exemple).

C.7 Les arcs dérivés du nœud "administration" portent les valeurs de code DCC (indicatif de pays pour la transmission de données) de la Recommandation X.121. Les arcs du niveau suivant sont définis selon les besoins par l'Administration du pays identifiée par le code DCC.

C.8 Les arcs dérivés du nœud "network-operator" portent les valeurs de code DNIC (code d'identification de réseau de données) de la Recommandation X.121. Les arcs du niveau suivant sont définis selon les besoins par l'Administration ou l'ER identifiée par le DNIC.

C.9 Les arcs dérivés du nœud "identified-organization" portent les valeurs qui leur sont affectées par le Bureau de normalisation des télécommunications (TSB) de l'UIT. Les arcs du niveau suivant sont définis selon les besoins par les organisations identifiées par la valeur affectée à l'arc antécédent.

NOTE – De tels arcs seront sans doute utiles aux travaux de divers types d'organismes, notamment:

- les ER n'exploitant pas de réseau public pour données;
- les organisations scientifiques et industrielles;
- les organisations de normalisation régionales;
- les organisations multinationales.

Annexe D

Affectation commune de valeurs de composantes d'identificateur d'objet

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

D.1 Trois arcs dérivent de la racine. L'affectation des valeurs et des identificateurs, ainsi que l'organisme habilité à affecter les valeurs de composantes qui en dérivent sont stipulés comme suit:

<i>Valeur</i>	<i>Identificateur</i>	<i>Organisme habilité à affecter les valeurs des composantes dérivées</i>
0	itu-t	UIT-T
1	iso	ISO
2	joint-iso-itu-t	Voir Annexe D

NOTE – La suite de cette annexe ne concerne que l'affectation de valeurs en commun par l'ISO et l'UIT-T.

D.2 Les identificateurs "itu-t", "iso" et "joint-iso-itu-t" affectés ci-dessus, peuvent chacun être utilisés comme une forme nominative "NameForm".

D.3 Les identificateurs "itu-t" et "joint-iso-itu-t" sont respectivement synonymes des identificateurs "ccitt" et "joint-iso-ccitt", et peuvent donc apparaître dans des valeurs d'identificateurs d'objets.

D.4 Les arcs dérivés du nœud "joint-iso-itu-t" portent des valeurs qui sont affectées et agréées périodiquement par l'ISO et l'UIT-T dans les domaines d'activité de normalisation conjoints de l'ISO et de l'UIT-T, conformément aux "Procédures d'affectation des valeurs de composantes d'identificateur d'objet pour utilisation commune par l'ISO et l'UIT-T"²⁾.

D.5 Les arcs issus de chacun des arcs établis par les mécanismes décrits au D.4 seront affectés conformément aux mécanismes institués lors de l'affectation de cet arc.

NOTE – Il est prévu que ceci impliquera une délégation de pouvoir, pour un agrément conjoint des éléments concernés par les Rapporteurs de l'UIT-T et de l'ISO dans le domaine commun de travail.

D.6 L'activité préalable à l'établissement de Normes internationales et de Recommandations dans les domaines d'activité conjoints de l'ISO et de l'UIT-T impose de réserver des identificateurs d'objets avant de suivre les procédures prévues au D.4, puis de les affecter conformément aux dispositions des Annexes B ou C. Les identificateurs des objets identifiés de cette manière par une Norme internationale ou une Recommandation ne devront pas être modifiés par les procédures décrites au D.4.

²⁾ L'autorité d'enregistrement chargée de l'affectation des valeurs de composantes d'identificateur d'objet pour une utilisation conjointe par l'ISO et l'UIT-T est l'ANSI (American National Standards Institute), 11 West 42nd Street, New York, NY 10036 USA.

Annexe E

Affectation de valeurs d'identificateurs d'objets

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

La présente Recommandation | Norme internationale affecte les valeurs suivantes:

Référence	Valeur d'identificateur d'objet
34.3	{ joint-iso-ccitt asn1(1) specification(0) characterStrings(1) numericString(0) }
	Valeur de descripteur d'objet
	"NumericString ASN.1 type"
Référence	Valeur d'identificateur d'objet
34.5	{ joint-iso-ccitt asn1(1) specification(0) characterStrings(1) printableString(1) }
	Valeur de descripteur d'objet
	"PrintableString ASN.1 type"
Référence	Valeur d'identificateur d'objet
35.1	{ joint-iso-ccitt asn1(1) specification(0) modules(0) iso10646(0) }
	Valeur de descripteur d'objet
	"ASN1 Character Module"

Annexe F

Exemples et conseils stylistiques

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

Cette annexe présente des exemples d'utilisation de la notation ASN.1 pour décrire des structures de données (fictives). Elle comporte également des conseils stylistiques et des directives concernant l'utilisation des diverses caractéristiques de la notation ASN.1. Sauf mention contraire, on suppose se trouver dans un environnement d'étiquetage automatique.

F.1 Exemple d'un enregistrement "salarié"

L'utilisation de la notation ASN.1 est illustrée par un enregistrement simple concernant un salarié fictif.

F.1.1 Description informelle de l'enregistrement "salarié"

La structure de l'enregistrement "salarié" est décrite ci-dessous, avec sa valeur pour un individu donné.

Nom:	John P. Smith
Fonction:	Directeur
Matricule:	51
Date d'embauche:	17 septembre 1971
Nom du conjoint:	Mary T. Smith
Nombre d'enfants:	2
Renseignements enfant	
Nom:	Ralph T. Smith
Date de naissance:	11 novembre 1957
Renseignements enfant	
Nom:	Susan B. Jones
Date de naissance:	17 juillet 1959

F.1.2 Description en notation ASN.1 de la structure de l'enregistrement

La structure de chaque enregistrement "salarié" est décrite formellement au moyen de la notation normalisée des types de données.

```

Enregistrement-salarié ::= [APPLICATION 0] SET
{
  nom           Nom,
  fonction      VisibleString,
  matricule     Matricule-salarié,
  date-embauche Date,
  nom-conjoint Nom,
  enfants       SEQUENCE OF Information-enfant DEFAULT { }
}

Information-enfant ::= SET
{
  nom           Nom,
  date-naissance Date
}

Nom ::= [APPLICATION 1] SEQUENCE
{
  prénom       VisibleString,
  initiale     VisibleString,
  nom-famille  VisibleString
}

Matricule-salarié ::= [APPLICATION 2] INTEGER

Date ::= [APPLICATION 3] VisibleString -- AAAA MMJJ

```

Cet exemple illustre un aspect d'analyse de la syntaxe ASN.1. La structure syntaxique "DEFAULT" ne peut être appliquée qu'à une composante d'une séquence ou d'un ensemble; elle ne peut pas être appliquée à une composante d'un type séquence-de. La déclaration "DEFAULT { }" de "Enregistrement-salarié" s'applique par conséquent à la composante "enfants" et non à la notation "Information-enfant".

F.1.3 Description en notation ASN.1 d'une valeur d'enregistrement

La valeur de l'enregistrement "salarié" de John P. Smith est décrite formellement ci-dessous en utilisant la notation normalisée des valeurs de données.

```
{ nom          {prénom "John", initiale "P", nom-famille "Smith"},
  fonction     "Directeur",
  matricule    51,
  date-embauche "19710917",
  nom-conjoint {prénom "Mary", initiale "T", nom-famille "Smith"},
  enfants
  { {nom {prénom "Ralph", initiale "T", nom-famille "Smith"},
    date-naissance "19571111"},
    {nom {prénom "Susan", initiale "B", nom-famille "Jones"},
      date-naissance "19590717" }
  }
}
```

F.2 Directives pour l'utilisation de la notation

La souplesse des types de données et de la notation formelle définis dans la présente Recommandation | Norme internationale permet de conceptualiser une grande variété de protocoles. Toutefois, cette souplesse peut parfois porter à confusion, en particulier lors d'une première utilisation de la notation. Cette annexe vise à minimiser les risques de confusion, en proposant des directives et des exemples concernant l'utilisation de la notation. Pour chacun des types de données prédéfinis, une ou plusieurs directives d'emploi sont proposées. Les types chaînes de caractères (les chaînes visibles par exemple) et les types définis dans les articles 39 à 41 ne sont pas traités ici.

F.2.1 Type booléen

F.2.1.1 Utiliser un type booléen pour modéliser les valeurs d'une variable logique (c'est-à-dire à deux états), par exemple la réponse par oui ou par non à une question.

EXEMPLE

Salarié ::= BOOLEAN

F.2.1.2 Pour affecter un nom de référence à un type booléen, en choisir un qui décrive l'état "Vrai".

EXEMPLE

Marié ::= BOOLEAN

et non

Situation de famille ::= BOOLEAN

F.2.2 Type entier

F.2.2.1 Utiliser un type entier pour modéliser les valeurs (de grandeur illimitée en pratique) d'une variable cardinale ou entière.

EXEMPLE

VérificationBalanceDesComptes ::= INTEGER -- *en centimes; négatif signifie découvert*
équilibre VérificationBalanceDesComptes ::= 0

F.2.2.2 Définir les valeurs minimale et maximale autorisées d'un type entier comme nombres nommés.

EXEMPLE

Quantième ::= INTEGER {premier(1), fin(31)}

aujourd'hui Quantième ::= premier

dateInconnue Quantième ::= 0

A noter que les nombres nommés "premier" et "fin" ont été choisis à cause de leur contenu sémantique pour le lecteur, et n'excluent pas la possibilité d'avoir des Quantièmes d'une valeur inférieure à 1, supérieure à 31 ou comprise entre 1 et 31.

Pour restreindre les valeurs de Quantième aux seules valeurs "premier" et "fin", il faudrait écrire:

Quantième ::= INTEGER {premier(1), fin(31)} (premier | fin)

et pour restreindre les valeurs de Quantième à toutes les valeurs comprises entre 1 et 31 inclusivement, il faudrait écrire:

Quantième ::= INTEGER {premier(1), fin(31)} (premier .. fin)

quantième Quantième ::= 4

F.2.3 Type énuméré

F.2.3.1 Utiliser un type énuméré pour modéliser les valeurs d'une variable qui possède trois états ou plus. Attribuer des valeurs en partant de zéro si leur seule contrainte est d'être distinctes.

EXEMPLE

**JourDeSemaine ::= ENUMERATED {dimanche(0), lundi(1), mardi(2),
mercredi(3), jeudi(4), vendredi(5), samedi(6)}**

débutDeSemaine JourDeSemaine ::= dimanche

A noter que les valeurs énumérées "dimanche", "lundi", etc., ont été choisies à cause de leur contenu sémantique pour le lecteur, et qu'à partir de ce moment, le type "JourDeSemaine" est restreint à ces seules valeurs. Ainsi, une valeur ne peut se voir affecter que les seuls noms "dimanche", "lundi", etc., et non par exemple les entiers équivalents.

F.2.3.2 Utiliser un type énuméré pour modéliser les valeurs d'une variable qui ne possède que deux états pour le moment, mais qui pourra en avoir d'autres dans une version ultérieure du protocole.

EXEMPLE

SituationDeFamille ::= ENUMERATED {célibataire(0), marié(1)}

en prévision de

SituationDeFamille ::= ENUMERATED {célibataire(0), marié(1), veuf(2)}

F.2.4 Type réel

F.2.4.1 Utiliser un type réel pour représenter un nombre irrationnel.

EXEMPLE

AngleEnRadians ::= REAL

pi REAL ::= {mantissa 3141592653589793238462643383279, base 10, exponent -30}

F.2.4.2 Les concepteurs d'applications peuvent souhaiter garantir une interopérabilité complète avec les valeurs réelles malgré les différences de représentation des nombres en virgule flottante dans les différents matériels, et décider par exemple d'utiliser des nombres en virgule flottante de longueur simple ou double dans une application X donnée. Ceci peut être réalisé de la manière suivante:

**App-X-Réel ::= REAL (WITH COMPONENTS {
mantissa (-16777215..16777215),
base (2),
exponent (-125..128) })**

-- Les expéditeurs ne transmettront pas de valeurs hors de ces intervalles;

-- les destinataires conformes seront capables de recevoir et de traiter

-- toutes les valeurs respectant ces limites d'intervalles.

circonférence App-X-Réel ::= {mantissa 16, base 2, exponent 1}

F.2.5 Type chaîne binaire

F.2.5.1 Utiliser une chaîne binaire pour modéliser des données binaires dont le format et la longueur ne sont pas spécifiés, ou sont spécifiés ailleurs, et dont la longueur n'est pas nécessairement un multiple de huit bits.

EXEMPLE

TélécopieG3 ::= BIT STRING

-- séquence binaire conforme à la Recommandation T.4

image TélécopieG3 ::= '100110100100001110110'B

boutDeChaîne BIT STRING ::= '0123456789ABCDEF'H

corps1 TélécopieG3 ::= '1101'B

corps2 TélécopieG3 ::= '1101000'B

A noter que "corps1" et "corps2" sont des valeurs abstraites différentes car les bits à 0 en fin de chaîne sont significatifs (en raison de l'absence d'une liste "NamedBitList" dans la définition de TélécopieG3).

F.2.5.2 Utiliser une chaîne binaire avec une contrainte de taille pour modéliser les valeurs d'un champ binaire de taille fixe.

EXEMPLE

ChampBinaire ::= BIT STRING (SIZE (12))

valeur1 ChampBinaire ::= '100110100100'B

valeur2 ChampBinaire ::= '9A4'H

valeur3 ChampBinaire ::= '1001101001'B -- *forme illicite: elle transgresse la contrainte de taille*

A noter que "valeur1" et "valeur2" sont une même valeur abstraite, les 4 bits en fin de "valeur2" ne sont pas significatifs.

F.2.5.3 Utiliser une chaîne binaire pour modéliser les valeurs d'un **champ d'indicateurs**, d'un ensemble ordonné de variables logiques correspondant à une collection ordonnée d'objets et indiquant chacun si une condition particulière est remplie par l'objet correspondant de la collection.

**JourDeSemaine ::= BIT STRING {
dimanche(0), lundi(1), mardi(2),
mercredi(3), jeudi(4), vendredi(5),
samedi(6) } (SIZE (0..7))**

joursEnsoleillésLaSemaine1 JourDeSemaine ::= {dimanche, lundi, mercredi}

joursEnsoleillésLaSemaine2 JourDeSemaine ::= '1101'B

joursEnsoleillésLaSemaine3 JourDeSemaine ::= '1101000'B

joursEnsoleillésLaSemaine4 JourDeSemaine ::= '11010000'B -- *forme illicite:
elle transgresse la contrainte de taille*

A noter que si la longueur de la chaîne binaire est inférieure à 7, cela signifie que les jours correspondant aux bits manquants ont été maussades; les trois premières valeurs ont donc la même valeur abstraite.

F.2.5.4 Utiliser une chaîne binaire pour modéliser les valeurs d'un **champ d'indicateurs**, d'un ensemble ordonné de taille fixe de variables logiques correspondant à une collection ordonnée d'objets et indiquant chacun si une condition particulière est remplie par l'objet correspondant de la collection.

**JourDeSemaine ::= BIT STRING {
dimanche(0), lundi(1), mardi(2),
mercredi(3), jeudi(4), vendredi(5),
samedi(6) } (SIZE (7))**

joursEnsoleillésSemaine1 JourDeSemaine ::= {dimanche, lundi, mercredi}

joursEnsoleillésSemaine2 JourDeSemaine ::= '1101'B -- *forme illicite:
elle transgresse la contrainte de taille*

joursEnsoleillésSemaine3 JourDeSemaine ::= '1101000'B

joursEnsoleillésSemaine4 JourDeSemaine ::= '11010000'B -- *forme illicite:
elle transgresse la contrainte de taille*

A noter que la première et la troisième valeur ont donc la même valeur abstraite.

F.2.5.5 Utiliser une chaîne binaire avec des bits nommés pour modéliser les valeurs d'un ensemble de variables logiques liées.

EXEMPLE

StatutPersonnel ::= BIT STRING

{marié(0), salarié(1), ancienCombattant(2), diplôméUniversitaire(3)}

billClinton StatutPersonnel ::= {marié, salarié, diplôméUniversitaire}

hillaryClinton StatutPersonnel ::= '110100'B

A noter que "billClinton" et "hillaryClinton" ont les mêmes valeurs abstraites.

F.2.6 Type chaîne d'octets

F.2.6.1 Utiliser un type chaîne d'octets pour modéliser des données binaires dont le format et la longueur ne sont pas spécifiés, ou sont spécifiés ailleurs, et dont la longueur est un multiple de huit bits.

- Le codage d'une ChaîneDCBCondensée ne comportera que le codage défini des caractères
- avec le champ de longueur nécessaire, ainsi que le champ étiquette en cas de codage selon
- les règles de codage de base BER. Les valeurs d'identificateurs d'objets ne sont pas
- transmises, la notation spécifiant le mode "fixed".

NOTE – Les règles de codage ne codent pas toujours les valeurs du type chaîne de caractères CHARACTER STRING sous une forme qui comporte nécessairement des valeurs d'identificateurs d'objets, bien que de tels identificateurs garantissent la préservation de la valeur abstraite dans le codage.

F.2.9 Type néant

Utiliser un type néant (NULL) pour indiquer l'absence effective d'un élément d'une séquence.

EXEMPLE

```

IdentificationMalade ::= SEQUENCE {
    nom VisibleString,
    numéroChambre CHOICE {
        chambre INTEGER,
        nonHospitalisé NULL -- s'il s'agit d'un malade non hospitalisé --
    }
}

dernierMalade IdentificationMalade ::= {
    nom "Jane Doe",
    numéroChambre nonHospitalisé : NULL
}
    
```

F.2.10 Types séquence et séquence-de

F.2.10.1 Utiliser un type séquence-de pour modéliser une collection de variables appartenant à un même type, en nombre élevé ou imprévisible, et dont l'ordre est significatif.

EXEMPLE

```

NomsDesPaysMembres ::= SEQUENCE OF VisibleString
-- par ordre alphabétique

deuxPremiers NomsDesPaysMembres ::= {"Australie", "Autriche"}
    
```

F.2.10.2 Utiliser un type séquence pour modéliser une collection de variables appartenant à un même type, en nombre limité et connu, et dont l'ordre est significatif, la composition de la collection ayant peu de chance de changer d'une version du protocole à la suivante.

EXEMPLE

```

NomDesMembresDuBureau ::= SEQUENCE {
    président VisibleString,
    vicePrésident VisibleString,
    secrétaire VisibleString}

cieAcme NomDesMembresDuBureau ::= {
    président "Jane Doe",
    vicePrésident "John Doe",
    secrétaire "Joe Doe"}
    
```

F.2.10.3 Utiliser un type séquence pour modéliser une collection de variables de types différents, en nombre connu et limité, et dont l'ordre est significatif, la composition de cette collection ayant peu de chance de changer d'une version du protocole à la suivante.

EXEMPLE

```

Habilitation ::= SEQUENCE {
    nomUtilisateur VisibleString
    motDePasse VisibleString
    numéroDeCompte INTEGER}
    
```

F.2.11 Types ensemble et ensemble-de

F.2.11.1 Utiliser un type ensemble pour modéliser une collection de variables en nombre connu et limité, et dont l'ordre n'est pas significatif. En l'absence d'étiquetage automatique, identifier chaque variable par un étiquetage spécifique au contexte comme le montre l'exemple suivant (en étiquetage automatique, les étiquettes ne sont pas nécessaires).

EXEMPLE

```

NomUtilisateur ::= SET {
    nomPersonnel      [0] VisibleString,
    nomDorganisation  [1] VisibleString,
    nomDePays         [2] VisibleString}

utilisateur NomUtilisateur ::= {
    nomDePays         "Nigéria",
    nomPersonnel      "Jonas Maruba",
    nomDorganisation  "Meteorology, Ltd."}

```

F.2.11.2 Utiliser un type ensemble avec déclaration OPTIONAL pour modéliser une collection de variables qui est un sous-ensemble (strict ou non strict) d'une autre collection de variables, dont le nombre est connu et raisonnablement petit, et dont l'ordre n'est pas significatif. En l'absence d'étiquetage automatique, identifier chaque variable par un étiquetage spécifique au contexte comme le montre l'exemple suivant (en étiquetage automatique, les étiquettes ne sont pas nécessaires).

EXEMPLE

```

NomUtilisateur ::= SET {
    nomPersonnel      [0] VisibleString,
    nomDorganisation  [1] VisibleString OPTIONAL,
    -- par défaut, nom de l'organisation locale --
    nomDePays         [2] VisibleString OPTIONAL
    -- par défaut, nom du pays local --}

```

F.2.11.3 Utiliser un type ensemble pour modéliser une collection de variables dont la composition risque de changer d'une version du protocole à la suivante. Identifier chaque variable par un étiquetage spécifique au contexte pour conserver le contrôle des étiquettes utilisées.

EXEMPLE

```

NomUtilisateur ::= SET {
    nomPersonnel      [0] VisibleString,
    nomDorganisation  [1] VisibleString OPTIONAL,
    -- par défaut, nom de l'organisation locale
    nomDePays         [2] VisibleString OPTIONAL
    -- par défaut, nom du pays local
    -- les autres attributs optionnels appellent un complément d'étude --}

utilisateur NomUtilisateur ::= {nomPersonnel "Jonas Maruba" }

```

F.2.11.4 Utiliser un type ensemble-de pour modéliser une collection de variables appartenant à un même type et dont l'ordre n'est pas significatif.

EXEMPLE

```

MotsClés ::= SET OF VisibleString -- en ordre arbitraire

quelquesMotsClésASN1 MotsClés ::= {"INTEGER", "BOOLEAN", "REAL"}

```

F.2.12 Type étiqueté

Avant l'introduction de la déclaration d'étiquetage automatique "AUTOMATIC TAGS", les spécifications ASN.1 comportaient souvent des étiquettes. Les points suivants décrivent la manière dont l'étiquetage était appliqué dans les cas types. Avec l'introduction de l'étiquetage automatique, l'utilisateur des spécifications ASN.1 n'a plus besoin d'utiliser la notation d'étiquettes, mais ceux qui apportent des modifications à des notations anciennes doivent en comprendre le fonctionnement.

F.2.12.1 Les étiquettes de la classe universelle ne sont utilisées que dans la présente Recommandation | Norme internationale. La notation [UNIVERSAL 30] (par exemple) n'est indiquée que pour assurer la précision de la définition des types utiles normalisés à l'échelle internationale. Elle ne doit pas être utilisée ailleurs.

F.2.12.2 Un style fréquent d'utilisation des étiquettes est d'affecter une étiquette de classe d'application exactement une fois à l'ensemble de la spécification, et de l'utiliser pour identifier un type qui s'avère être largement utilisé un peu partout dans la spécification. On utilise souvent aussi une étiquette de classe d'application (une seule fois) pour étiqueter les types dans la structure choisie la plus externe d'une application, en assurant l'identification des différents messages par l'étiquette de classe d'application. Ce dernier cas est illustré dans l'exemple ci-dessous:

EXEMPLE

```
NomDeFichier ::= [APPLICATION 8] SEQUENCE {
    nomDeRépertoire      VisibleString,
    nomDeFichierSimple   VisibleString}
```

F.2.12.3 L'étiquetage propre au contexte est fréquemment appliqué de manière algorithmique à toutes les composantes d'une structure ensemble, séquence ou choix. A noter toutefois que l'étiquetage automatique décharge le concepteur de cette tâche.

EXEMPLE

```
EnregistrementClient ::= SET {
    nom           [0] VisibleString,
    adressePostale [1] VisibleString,
    numéroDeCompte [2] INTEGER,
    solde         [3] INTEGER -- en centimes --}

AttributClient ::= CHOICE {
    nom           [0] VisibleString,
    adressePostale [1] VisibleString,
    numéroDeCompte [2] INTEGER,
    solde         [3] INTEGER -- en centimes --}
```

F.2.12.4 L'étiquetage en classe privée ne devrait normalement pas être utilisé dans les Spécifications internationales (bien que ceci ne puisse être interdit). Les applications développées par les entreprises devraient normalement utiliser les classes d'étiquetage d'application et propre au contexte. Mais une situation particulière peut se présenter dans laquelle une spécification propre à une entreprise apporte une extension à une Spécification internationale; dans une telle circonstance, l'utilisation d'étiquettes de la classe privée a l'avantage de protéger partiellement la spécification propre à l'entreprise d'une modification de la Spécification internationale.

EXEMPLE

```
NuméroDeBadgeAcme ::= [PRIVATE 2] INTEGER

numéroDeBadge NuméroDeBadgeAcme ::= 2345
```

F.2.12.5 L'ajout de la déclaration "IMPLICIT" à chaque étiquette n'est généralement trouvé que sur les spécifications les plus anciennes. Les règles de codage de base (BER) produisent une représentation moins compacte avec l'étiquetage explicite qu'avec l'étiquetage implicite. Les règles de codage en paquet (PER) aboutissent à la même compacité de codage dans les deux cas. Avec les règles BER et l'étiquetage explicite, il y a une meilleure visibilité du type sous-jacent (entier, réel, booléen, etc.) dans les données codées. Les présentes directives utilisent l'étiquetage implicite dans les exemples chaque fois qu'il est licite de le faire. Selon les règles de codage, cela peut aboutir à une représentation compacte, particulièrement recherchée dans certaines applications. Dans d'autres applications, la compacité peut être moins importante par exemple que la possibilité d'effectuer une solide vérification de type. Dans une telle situation, on peut recourir à l'étiquetage explicite.

EXEMPLE

```
EnregistrementClient ::= SET {
    nom           [0] IMPLICIT VisibleString,
    adressePostale [1] IMPLICIT VisibleString,
    numéroDeCompte [2] IMPLICIT INTEGER,
    solde         [3] IMPLICIT INTEGER -- en centimes --}

AttributClient ::= CHOICE {
    nom           [0] IMPLICIT VisibleString,
    adressePostale [1] IMPLICIT VisibleString,
    numéroDeCompte [2] IMPLICIT INTEGER,
    solde         [3] IMPLICIT INTEGER -- en centimes --}
```

F.2.12.6 Les directives d'étiquetage pour les nouvelles spécifications ASN.1 d'utilisateur faisant référence à la présente Recommandation | Norme internationale sont très simples: NE PAS ÉTIQUETER, METTRE LA DÉCLARATION "AUTOMATIC TAGS" dans l'en-tête du module, et ne plus s'en occuper. S'il s'avère nécessaire d'ajouter une nouvelle composante à une structure ensemble, séquence ou choix dans une version ultérieure, l'ajouter à la fin.

F.2.13 Type choix

F.2.13.1 Utiliser un type choix pour modéliser une variable choisie dans une collection de variables en nombre connu et limité.

EXEMPLE

```
IdentificateurDeFichier ::= CHOICE {
    nomRelatif      VisibleString,
    -- nom du fichier (par exemple, "RapportDavancementDeMars")
    nomAbsolu       VisibleString,
    -- noms du fichier et du répertoire qui le contient
    -- (par exemple, "<Williams>RapportDavancementDeMars")
    numéroDeSérie   INTEGER
    -- identificateur affecté par le système au fichier -- }
```

fichier IdentificateurDeFichier ::= numéroDeSérie : 106448503

F.2.13.2 Utiliser un type choix pour modéliser une variable choisie dans une collection de variables dont la composition risque de changer d'une version du protocole à la suivante.

EXEMPLE

```
IdentificateurDeFichier ::= CHOICE {
    nomRelatif      VisibleString,
    -- nom du fichier (par exemple, "RapportDavancementDeMars")
    nomAbsolu       VisibleString
    -- noms du fichier et du répertoire qui le contient
    -- (par exemple, "<Williams>RapportDavancementDeMars")
    -- d'autres formes d'identificateurs de fichier appellent un complément d'étude -- }
```

F.2.13.3 Lorsque l'étiquetage implicite est la règle dans une application particulière de la présente Recommandation | Norme internationale, utiliser un choix d'un seul type lorsqu'il est envisagé d'avoir ultérieurement plusieurs types possibles.

EXEMPLE

```
Voeux ::= [APPLICATION 12] CHOICE {
    cartePostale    VisibleString}
```

en prévision de

```
Voeux ::= [APPLICATION 12] CHOICE {
    cartePostale    VisibleString,
    enregistrement Voix}
```

F.2.13.4 Utiliser plusieurs signes "deux points" lorsqu'un type choix est imbriqué dans un autre type choix.

EXEMPLE

```
Voeux ::= [APPLICATION 12] CHOICE {
    cartePostale    VisibleString,
    enregistrement Voix}
```

```
Voix ::= CHOICE {
    anglais    OCTET STRING,
    swahili    OCTET STRING}
```

mesVoeux Voeux ::= enregistrement : anglais : '019838547E0'H

F.2.14 Type sélection

F.2.14.1 Utiliser un type sélection pour modéliser une variable dont le type est celui d'une des formes particulières d'un choix défini antérieurement.

F.2.14.2 Considérons la définition:

```
AttributFichier ::= CHOICE {
    date-dernière-utilisation    INTEGER,
    nom-fichier                  VisibleString}
```

La définition suivante est alors possible:

```
ListeDesAttributs ::= SEQUENCE {
    premier-attribut    date-dernière-utilisation < AttributFichier,
    second-attribut    nom-fichier < AttributFichier}
```

avec la notation de valeur possible suivante:

```
listeDattributs ListeDesAttributs ::= {
    premier-attribut    27,
    second-attribut    "PROGRAMME" }
```

F.2.15 Type champ de classe d'objets

F.2.15.1 Utiliser un type champ de classe d'objets pour identifier un type défini au moyen d'une classe d'objets informationnels (voir la Rec. UIT-T X.681 | ISO/CEI 8824-2). Par exemple, les champs de la classe d'objets informationnels ATTRIBUT peuvent être utilisés pour définir un type "Attribut".

EXEMPLE

```
ATTRIBUT ::= CLASS
{
    &TypeDeLattribut,
    &idDeLattribut    OBJECT IDENTIFIER UNIQUE
}
Attribut ::= SEQUENCE {
    idDattribut    ATTRIBUT.&idDeLattribut,    -- normalement, une contrainte est imposée
    valeurDattribut    ATTRIBUT.&TypeDeLattribut    -- normalement, une contrainte est imposée
}
```

"ATTRIBUT.&idDeLattribut" et "ATTRIBUT.&TypeDeLattribut" sont tous deux des types de champ de classe d'objets, en ce sens qu'il s'agit de types définis par référence à une classe d'objets informationnels (ATTRIBUT). Le type "ATTRIBUT.&idDeLattribut" est fixé parce qu'il est explicitement défini dans ATTRIBUT comme identificateur d'objet OBJECT IDENTIFIER. Par contre, le type "ATTRIBUT.&TypeDeLattribut" peut contenir une valeur de n'importe quel type défini en ASN.1, puisque son type n'est pas fixé dans la définition de la classe ATTRIBUT. Une notation qui possède cette propriété de pouvoir véhiculer une valeur de n'importe quel type est dite "notation de type ouvert". Ainsi, "ATTRIBUT.&TypeDeLattribut" est un type ouvert.

F.2.16 Type pdv encapsulé

F.2.16.1 Utiliser un type pdv encapsulé pour modéliser une variable dont le type n'est pas spécifié, ou l'est ailleurs sans restriction quant à la notation utilisée pour le spécifier.

EXEMPLE

```
ContenuFichier ::= EMBEDDED PDV
ListeDeDocuments ::= SEQUENCE OF EMBEDDED PDV
```

F.2.17 Type externe

Le type externe est similaire au type pdv encapsulé, mais dispose d'un nombre plus restreint de possibilités d'identification. On préférera généralement dans les nouvelles spécifications utiliser le type pdv encapsulé à cause de sa plus grande souplesse et parce que certaines règles de codage en codent les valeurs de manière plus efficace.

F.2.18 Type instance-de

F.2.18.1 Utiliser une déclaration instance-de pour spécifier un type contenant un champ identificateur d'objet et un type ouvert dont la valeur appartient à un type déterminé par cet identificateur d'objet. Les types instance-de sont astreints à véhiculer une valeur de la classe identificateur de type TYPE-IDENTIFIER (voir les Annexes A et C de la Rec. UIT-T X.681 | ISO/CEI 8824-2).

EXEMPLE

CLASSE-CONTROLE-D-ACCES ::= TYPE-IDENTIFIER

```

Invocation ::= SEQUENCE {
  classeDobjet      ClasseDobjet,
  instanceDobjet   InstanceDobjet,
  controleDacces   INSTANCE OF CLASSE-CONTROLE-D-ACCES,
                    -- normalement, une contrainte est imposée
  idDattribut     ATTRIBUT.&idDeLattribut
}

```

L'invocation est alors équivalente à:

```

Invocation ::= SEQUENCE {
  classeDobjet      ClasseDobjet,
  instanceDobjet   InstanceDobjet,
  controleDacces   [UNIVERSAL 8] IMPLICIT SEQUENCE {
    id-de-type      CLASSE-CONTROLE-D-ACCES.&id,
                    -- normalement, une contrainte est imposée
    valeur          [0] CLASSE-CONTROLE-D-ACCES.&Type
                    -- normalement, une contrainte est imposée
  },
  idDattribut     ATTRIBUT.&idDeLattribut
}

```

La véritable utilité du type instance-de n'apparaît que lorsqu'il est contraint au moyen d'un ensemble d'objets informationnels, mais un tel exemple sort du cadre de la présente Recommandation | Norme internationale. Se reporter à la Rec. UIT-T X.682 | ISO/CEI 8824-3 pour la définition de l'ensemble d'objets informationnels, et à l'Annexe A du même document pour la manière d'utiliser un ensemble d'objets informationnels pour contraindre un type instance-de. A noter que le codage d'une instance-de CLASSE-CONTROLE-D-ACCES est le même que celui obtenu par une valeur de type externe n'ayant qu'un identificateur d'objet et une valeur de données.

F.3 Identification des syntaxes abstraites

F.3.1 L'utilisation du service de présentation défini dans la Rec. UIT-T X.216 | ISO/CEI 8822 impose de spécifier des valeurs appelées valeurs de données de présentation et le groupage de ces valeurs en des ensembles appelés syntaxes abstraites. Chacun de ces ensembles reçoit un nom de syntaxe abstraite du type ASN.1 identificateur d'objet.

F.3.2 L'ASN.1 peut servir d'outil général pour la spécification des valeurs de données de présentation et leur groupage en syntaxes abstraites nommées.

F.3.3 Dans le cas le plus simple d'une telle utilisation, il existe un seul type ASN.1 et chaque valeur de données de présentation de la syntaxe abstraite nommée est une valeur de ce type ASN.1. Normalement, il s'agira d'un type choix, et chaque valeur de données de présentation sera une forme possible de ce type choix. Dans ce cas, il est recommandé d'utiliser la notation de module ASN.1 avec ce type choix défini en premier, suivi de la définition des types (non universels) auxquels le type choix se réfère directement ou non.

NOTE – Cette disposition n'exclut pas la référence à des types définis dans d'autres modules.

F.3.4 Il est recommandé que l'affectation d'un identificateur d'objet et d'un descripteur d'objet à une syntaxe abstraite soit effectuée au moyen de l'utile classe d'objets informationnels ABSTRACT-SYNTAX définie dans la Rec. UIT-T X.681 | ISO/CEI 8824-2. Il est également recommandé que toutes les utilisations de la classe ABSTRACT-SYNTAX soient regroupées dans un même module "racine" identifiant toutes les syntaxes abstraites utilisées par une norme d'application.

F.3.5 Ce qui suit est un exemple de texte pouvant apparaître dans une norme d'application.

EXEMPLE

```

ISOxxxx-yyyy {iso standard xxxx asn1-modules(...)} yyyy-pdu(...)} DEFINITIONS ::=
BEGIN
  EXPORTS YYYYY-PDU;
  YYYYY-PDU ::= CHOICE {
    connect-pdu ..... ,
    data-pdu CHOICE {

```

```

        .... ,
        ....
    },
    ....
}
.....
END

ISOxxxx-yyyy-Abstract-Syntax-Module {iso standard xxxx asn1-modules(...)} DEFINITIONS ::=
BEGIN
    IMPORTS YYYYY-PDU FROM ISOxxxx-yyyy {iso standard xxxx asn1-modules(...)} yyyyy-pdu(...);

-- La présente Recommandation | Norme internationale définit la syntaxe abstraite suivante:

    YYYYY-Abstract-Syntax ABSTRACT-SYNTAX ::=
        { YYYYY-PDU IDENTIFIED BY yyyyy-abstract-syntax-object-id }

    yyyyy-abstract-syntax-object-id OBJECT IDENTIFIER ::= {iso standard yyyyy(xxxx) abstract-syntax(...)}

-- Le descripteur d'objet correspondant est:

    yyyyy-abstract-syntax-descriptor ObjectDescriptor ::= "....."

-- Les valeurs d'identificateur d'objet et de descripteur d'objet ASN.1:
    ..... -- identificateur d'objet de règles de codage
    ..... -- descripteur d'objet de règles de codage
-- affectées aux règles de codage dans les Rec. UIT-T X.690 - 691 | ISO/CEI 8825-1 - 2
-- peuvent être utilisées comme identificateur de la syntaxe de
-- transfert en conjonction avec la syntaxe de transfert présente ISOxxxx-yyyy-Abstract-Syntax.

END

```

F.3.6 Pour garantir l'interopérabilité, la norme peut rendre de surcroît obligatoire la prise en charge de la syntaxe de transfert obtenue en appliquant les règles de codage mentionnées dans son module de syntaxe abstraite.

F.4 Sous-types

F.4.1 Utiliser des sous-types pour restreindre un type existant à des valeurs particulières.

EXEMPLE

```

NuméroAtomique ::= INTEGER (1..104)

NuméroTéléphoniqueAuClavier ::= IA5String
    (FROM ("0123456789" | "*" | "#" )) (SIZE (1..63))

ListeDesParamètres ::= SET SIZE (1..63) OF Paramètres

PetitPremier ::= INTEGER (2|3|5|7|11|13|17|19|23|29)

```

F.4.2 Lorsque deux types apparentés ou plus ont de nombreux points communs, envisager de définir explicitement leur parent commun comme un type, et d'en dériver chacun des types par sous-typage. Cette solution permet de mettre en relief leurs relations et leurs points communs et de maintenir cette précision (sans contrainte) au fur et à mesure que les types évoluent. Elle facilite donc l'adoption d'approches communes pour le traitement des valeurs de ces types.

EXEMPLE

```

Enveloppe ::= SET {
    typeA TypeA,
    typeB TypeB OPTIONAL,
    typeC TypeC OPTIONAL}
-- le parent commun

ABEnveloppe ::= Enveloppe (WITH COMPONENTS
    {... ,
    typeB PRESENT, typeC ABSENT})
-- le typeB devant toujours apparaître mais pas le typeC

```

```

ACEnveloppe ::= Enveloppe (WITH COMPONENTS
    { ... ,
      typeB ABSENT, typeC PRESENT })
-- le typeC devant toujours apparaître mais pas le typeB

```

Ces dernières définitions peuvent aussi être exprimées de la manière suivante:

```

ABEnveloppe ::= Enveloppe (WITH COMPONENTS {typeA, typeB})
ACEnveloppe ::= Enveloppe (WITH COMPONENTS {typeA, typeC})

```

Le choix entre les différentes formes dépend de facteurs comme le nombre de composantes du type parent et, parmi celles-ci, le nombre de celles qui sont optionnelles, l'ampleur des différences entre les différents types et la stratégie d'évolution probable.

F.4.3 Recourir au sous-typage pour définir partiellement une valeur, lorsqu'une unité de données protocolaire par exemple doit être soumise à un test de conformité ne portant que sur certaines de ses composantes.

EXEMPLE

Soit:

```

PDU ::= SET
{alpha INTEGER,
 beta IA5String OPTIONAL,
 gamma SEQUENCE OF Parameter,
 delta BOOLEAN}

```

alors, pour établir un test exigeant que le booléen soit faux et l'entier négatif, écrire:

```

TestPDU ::= PDU (WITH COMPONENTS
    { ... ,
      delta (FALSE),
      alpha (MIN..<0)})

```

et si, de plus, la chaîne IA5, beta, doit être présente et longue de 5 ou 12 caractères, écrire:

```

PDUtestSupplémentaire ::= PDUtest (WITH COMPONENTS { ... , beta (SIZE (5|12)) PRESENT })

```

F.4.4 Si un type de données d'usage général a été défini comme un type séquence-de, recourir au sous-typage pour définir un sous-type restreint de ce type général:

EXEMPLE

```

Bloc-texte ::= SEQUENCE OF VisibleString
Adresse ::= Bloc-texte (SIZE (1..6)) (WITH COMPONENT (SIZE (1..32)))

```

F.4.5 Si un type de données d'usage général a été défini comme un type CHOIX, recourir au sous-typage pour définir un sous-type restreint de ce type général:

EXEMPLE

```

Z ::= CHOICE {
  a A,
  b B,
  c C,
  d D,
  e E
}

```

```

V ::= Z (WITH COMPONENTS { ..., a ABSENT, b ABSENT }) -- 'a' et 'b' doivent être absents, et soit 'c',
-- soit 'd', soit 'e' peut être présent dans
-- la valeur

```

```

W ::= Z (WITH COMPONENTS { ..., a PRESENT }) -- seul 'a' doit être présent (voir 45.8.9.2)

```

```

X ::= Z (WITH COMPONENTS { a PRESENT }) -- seul 'a' doit être présent (voir 45.8.9.2)

```

```

Y ::= Z (WITH COMPONENTS { a ABSENT, b, c }) -- 'a', 'd' et 'e' doivent être absents, et soit 'b',
-- soit 'c' peut être présent dans la valeur

```

NOTE – W et X sont sémantiquement identiques.

F.4.6 Utiliser des sous-types contenus pour former de nouveaux sous-types à partir de sous-types existants:

EXEMPLE

```
Mois ::= ENUMERATED {
    janvier      (1),
    février     (2),
    mars        (3),
    avril       (4),
    mai         (5),
    juin        (6),
    juillet     (7),
    août         (8),
    septembre  (9),
    octobre    (10),
    novembre   (11),
    décembre   (12) }
```

```
Premier-trimestre ::= Mois (
    janvier |
    février |
    mars )
```

```
Deuxième-trimestre ::= Mois (
    avril |
    mai |
    juin )
```

```
Troisième-trimestre ::= Mois (
    juillet |
    août |
    septembre )
```

```
Quatrième-trimestre ::= Mois (
    octobre |
    novembre |
    décembre )
```

```
Premier-semestre ::= Mois ( Premier-trimestre | Deuxième-trimestre )
```

```
Deuxième-semestre ::= Mois ( Troisième-trimestre | Quatrième-trimestre )
```

Annexe G

Annexe didactique sur les chaînes de caractères ASN.1

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

G.1 Prise en charge des chaînes de caractères en notation ASN.1

G.1.1 La notation ASN.1 prend en charge quatre groupes de chaînes de caractères, à savoir:

- a) les types de chaînes de caractères basés sur le *Registre international des jeux de caractères codés à utiliser avec les séquences d'échappement de l'ISO* (c'est-à-dire la structure de ISO 646) et le Registre international associé des jeux de caractères codés, constitués par les types de chaîne visible VisibleString, AI n° 5 IA5String, télétext TeletexString, vidéo VideotexString, graphique GraphicString et générale GeneralString;
- b) les types de chaînes de caractères basés sur ISO/CEI 10646-1, obtenus par restriction des types de chaîne universelle UniversalString ou multilingue BMPString à des sous-ensembles définis dans ISO/CEI 10646-1 ou en utilisant des caractères nommés;

NOTES

1 L'utilisation du type de chaîne universelle UniversalString non restreinte conduit à une transgression des prescriptions de conformité concernant l'échange d'informations telles qu'elles sont spécifiées dans ISO/CEI 10646-1 car aucun sous-ensemble adopté n'a été spécifié.

2 Malgré ce qui précède, l'utilisation de ce type avec une simple contrainte de sous-typage utilisant un paramètre de la syntaxe abstraite (restreinte à un sous-type défini du type UniversalString) peut constituer un outil puissant et souple pour le traitement des caractères en s'appuyant sur des profils pour déterminer la valeur du paramètre permettant de répondre aux exigences d'un domaine d'intérêt commun. Cependant, l'utilisation de la déclaration CHARACTER STRING doit généralement être préférée dans les Recommandations | Normes internationales (voir ci-dessous).

- c) les types de chaînes de caractères fournissant une petite collection simple de caractères, spécifiés dans la présente Recommandation | Norme internationale, et destinés à des utilisations particulières; il s'agit des types de chaîne numérique NumericString et imprimable PrintableString;
- d) le type CHARACTER STRING utilisé avec négociation du jeu de caractères à utiliser (ou l'annonce du jeu utilisé); cette possibilité permet à une application d'utiliser tous les jeux de caractères et codages auxquels des identificateurs d'objets OBJECT IDENTIFIER ont été affectés, y compris ceux du *Registre international des jeux de caractères codés à utiliser avec les séquences d'échappement de l'ISO*, de ISO 7350 et ISO/CEI 10646-1, et les jeux de caractères et codages privés; (les profils peuvent imposer des spécifications ou des restrictions aux jeux de caractères – les syntaxes abstraites de caractères – à utiliser).

G.2 Les types chaîne universelle "UniversalString" et table multilingue "BMPString"

G.2.1 Le type chaîne universelle "UniversalString" contient tous les caractères définis dans ISO/CEI 10646-1. Ce jeu est généralement trop grand pour imposer en pratique une contrainte de conformité, et il doit normalement être restreint à des sous-ensembles combinant les jeux de caractères normalisés définis en Annexe A de ISO/CEI 10646-1.

G.2.2 Le type chaîne de table multilingue "BMPString" contient tous les caractères de la grille ISO/CEI 10646-1 (les premiers 64K caractères). Ce jeu est généralement restreint à une combinaison des jeux de caractères normalisés définis en Annexe A de ISO/CEI 10646-1.

G.2.3 Il existe pour les jeux définis en Annexe A de ISO/CEI 10646-1 des références de type définies dans le module prédéfini ASN1-CHARACTER-MODULE (voir article 35). Le mécanisme de contrainte de sous-typage permet de définir de nouveaux sous-types de UniversalString par combinaison de sous-types existants.

G.2.4 Parmi les références de type définies dans le module ASN1-CHARACTER-MODULE et le nom de collection correspondant dans ISO/CEI 10646-1, on peut citer à titre d'exemple:

BasicLatin	LATIN DE BASE
Latin-1Supplement	SUPPLÉMENT LATIN-1
LatinExtended-A	LATIN ÉTENDU-A
LatinExtended-B	LATIN ÉTENDU-B

IpaExtensions	EXTENSIONS IPA
SpacingModifierLetters	LETTRES MODIFIANT LES ESPACEMENTS
CombiningDiacriticalMarks	SIGNES DIACRITIQUES DE COMBINAISON

G.2.5 ISO/CEI 10646-1 spécifie trois "niveaux d'application", et impose dans toute utilisation de ISO/CEI 10646-1 de spécifier le niveau de l'application.

Le niveau d'application a trait à l'étendue de la prise en charge des *caractères de combinaison* dans le répertoire de caractères, et par là, en termes ASN.1, il définit un sous-ensemble des types de chaînes de caractères restreintes universelle UniversalString et multilingue BMPString.

Dans le niveau 1 de mise en œuvre, les caractères de combinaison sont interdits, et il existe normalement une correspondance biunivoque entre les caractères abstraits (références de cellules) des chaînes de caractères ASN.1 et les caractères matériels imprimés ou affichés de la chaîne.

Dans le niveau 2 de mise en œuvre, il est possible d'utiliser certains caractères de combinaison (énumérés en Annexe B de ISO/CEI 10646-1), mais d'autres sont interdits.

Dans le niveau 3 de mise en œuvre, il n'y a aucune restriction à l'utilisation des caractères de combinaison.

G.3 A propos des prescriptions de conformité à ISO/CEI 10646-1

L'utilisation d'un type (ou d'un sous-type de) chaîne universelle "UniversalString" (cellules codées sur 4 octets) ou chaîne de table multilingue "BMPString" (cellules codées sur 2 octets) dans une définition de type ASN.1 impose d'étudier les prescriptions de conformité de ISO/CEI 10646-1.

Ces prescriptions de conformité imposent aux développeurs d'une certaine norme (disons la norme X) utilisant de tels types ASN.1 de produire une déclaration (dans la déclaration PICS de conformité d'une instance de protocole) précisant le sous-ensemble de caractères de ISO/CEI 10646-1 qu'ils utilisent dans leur instance de la norme X, ainsi que le niveau d'application (prise en charge des caractères de combinaison) de cette instance.

L'utilisation d'un sous-type de "UniversalString" ou de "BMPString" dans une spécification impose la prise en charge par l'instance de tous les caractères ISO/CEI 10646-1 inclus dans ce sous-type ASN.1, et donc que ces caractères (au moins) soient présents dans le sous-ensemble adopté pour l'instance de protocole. De même, il est nécessaire que le niveau de mise en œuvre déclaré soit pris en charge par tous les sous-types ASN.1 ainsi définis.

NOTE – En l'absence de paramètres dans la syntaxe abstraite et de spécifications d'exceptions, une spécification ASN.1 détermine à la fois le jeu (maximal) de caractères pouvant être utilisé en émission et le jeu (minimal) de caractères qui doit pouvoir être traité en réception. L'adoption du jeu ISO/CEI 10646-1 signifie qu'aucun caractère en dehors de ce jeu ne doit être transmis et que tous les caractères de ce jeu soient pris en charge à la réception. Le jeu adopté doit donc correspondre très précisément à l'ensemble de tous les caractères autorisés par la spécification ASN.1. Le cas où intervient un paramètre de syntaxe abstraite est discuté ci-dessous.

G.4 Recommandations aux utilisateurs ASN.1 à propos de la conformité à ISO/CEI 10646-1

Les utilisateurs ASN.1 doivent indiquer clairement le jeu de caractères ISO/CEI 10646-1 (et le niveau d'application correspondant) qui devra être adopté par les instances de protocole pour que celles-ci puissent être conformes aux prescriptions de leur norme.

Une bonne façon de le faire est de définir un sous-type ASN.1 du type de "UniversalString" ou "BMPString" contenant tous les caractères nécessaires à la norme, et d'en restreindre s'il y a lieu le niveau au niveau 1 ou 2. On pourra par exemple désigner ce type par "ISO-10646-String".

EXEMPLE

ISO-10646-String ::= BMPString

(FROM(Level2 INTERSECTION (BasicLatin UNION HebrewExtended UNION Hiragana)))

-- Il s'agit du type qui définit le jeu minimal de caractères dans la collection

-- adoptée pour les instances de cette norme.

-- Le niveau d'application requis est le niveau 2 au moins.

La déclaration PICS contiendrait alors une simple déclaration indiquant que le sous-ensemble de caractères ISO/CEI 10646-1 adopté est le jeu (et le niveau) limité défini par le type "ISO-10646-String", et ce type (ou un sous-type de) "ISO-10646-String" sera utilisé partout dans la norme où apparaîtront des chaînes de caractères ISO/CEI 10646-1.

EXEMPLE DE DÉCLARATION PICS

Le sous-jeu de caractères ISO/CEI 10646-1 adopté est le sous-ensemble limité constitué de tous les caractères du type ASN.1 "ISO-10646-String" défini dans le module <mettez ici le nom de votre module>, avec un niveau d'application de 2.

EXEMPLE D'UTILISATION DANS UN PROTOCOLE

```

Message ::= SEQUENCE {
    premier-champ    ISO-10646-String,
                    -- tous les caractères du sous-jeu peuvent être utilisés
    deuxième-champ  ISO-10646-String (FROM (latinSmallLetterA .. latinSmallLetterZ)),
                    -- minuscules latines seulement
    troisième-champ ISO-10646-String (FROM (digitZero .. digitNine))
                    -- chiffres seulement
}

```

G.5 Sous-jeux adoptés comme paramètres de la syntaxe abstraite

ISO/CEI 10646-1 impose de définir explicitement le sous-jeu de caractères et le niveau d'application adoptés. Lorsqu'un utilisateur ASN.1 ne souhaite pas imposer une contrainte au jeu de caractères ISO/CEI 10646-1 dans une partie quelconque de la norme qu'il définit, il peut l'exprimer en définissant "ISO-10646-String" (par exemple) comme un sous-type de chaîne universelle (codage sur 4 octets) ou de chaîne multilingue (codage sur 2 octets) avec une contrainte de sous-typage constituée de (ou comportant) un type "ImplementorsSubset" qui est laissé comme paramètre de la syntaxe abstraite.

Les utilisateurs de l'ASN.1 doivent savoir que dans ce cas, un expéditeur conforme peut transmettre à un destinataire conforme des caractères qui ne pourront pas être traités par le destinataire parce qu'ils n'appartiennent pas au sous-jeu ou au niveau d'application (dépendants de l'application) adoptés par le destinataire; il est recommandé dans ce cas d'inclure dans la définition du type "ISO-10646-String" une spécification de traitement d'exception.

EXEMPLE

```

ISO-10646-String {UniversalString : SousJeuApplication, NiveauApplication} ::=
    UniversalString (FROM((SousJeuApplication UNION BasicLatin)
                        INTERSECTION NiveauApplication) !ProblèmeJeuDeCaractères)
    -- Le sous-jeu pris dans ISO/CEI 10646-1 comprendra le jeu "BasicLatin", mais pourra
    -- également inclure tout caractère spécifié dans "SousJeuApplication", qui est un
    -- paramètre de la syntaxe abstraite. "NiveauApplication", qui est aussi un paramètre de
    -- la syntaxe abstraite, définit le niveau de l'application. Un destinataire conforme devra
    -- être préparé à recevoir des caractères n'appartenant pas au sous-jeu de caractères et
    -- au niveau de son application. Dans un tel cas, le système invoquera le traitement
    -- d'exception spécifié au paragraphe <ajouter ici votre numéro de paragraphe> pour
    -- "ProblèmeJeuDeCaractères". A noter qu'un tel traitement ne sera jamais invoqué par
    -- un destinataire conforme si les caractères utilisés dans la communication sont limités
    -- au jeu "BasicLatin".

```

```

Mon-Type-Chaîne-de-Niveau-2 ::= ISO-10646-String { { HebrewExtended UNION Hiragana }, Level2 }

```

G.6 Le type chaîne de caractères CHARACTER STRING

G.6.1 Le type CHARACTER STRING offre une totale souplesse de choix du jeu de caractères et de la méthode de codage. Lorsqu'une connexion simple (sans application relais) assure un transfert de données de bout en bout, la négociation des jeux de caractères à utiliser et du codage peut s'effectuer dans le cadre de la définition des contextes de présentation pour les syntaxes abstraites de caractères.

G.6.2 Il est important de comprendre qu'une syntaxe abstraite de caractères est une syntaxe abstraite ordinaire avec des restrictions apportées à ses valeurs possibles (ce sont toutes des chaînes de caractères formées à partir d'une certaine collection de caractères). L'enregistrement de telles syntaxes, ainsi que la négociation d'un contexte de présentation, s'effectue donc de manière normale.

G.6.3 Le codage du type CHARACTER STRING permet aussi de déclarer sans négociation les syntaxes abstraites et de transfert utilisées dans des environnements où il convient de procéder de la sorte.

NOTES

1 Les concepteurs d'applications peuvent interdire ou imposer la négociation d'une présentation pour ces champs, ou en laisser le choix à l'expéditeur.

2 Lorsqu'il donne la préférence au mode de déclaration plutôt que de négociation, le concepteur de l'application doit examiner à la fois comment l'expéditeur peut déterminer les syntaxes abstraites (et de transfert) de caractères acceptables pour le destinataire (en ayant recours par exemple au service d'Annuaire ou en utilisant la technique des profils), et aussi les mesures qu'un destinataire doit prendre s'il reçoit une valeur de type CHARACTER STRING provenant d'une syntaxe abstraite de caractères qu'il ne prend pas en charge.

G.6.4 Si la négociation est utilisée, le concepteur de la couche application peut contrôler cette négociation en spécifiant les moments où de tels contextes de présentation doivent être établis et en spécifiant le paramètre "données d'utilisateur" des primitives de modification de contexte de Présentation P-ALTER-CONTEXT, ou peut simplement supposer qu'un profil donné aura déterminé la syntaxe abstraite à utiliser, établissant un contexte de présentation pour celle-ci au moment de la connexion de Présentation P-CONNECT.

G.6.5 Les fonctionnalités de gestion du contexte de service de présentation permettent à l'initiateur de la connexion de proposer (dans une primitive P-CONNECT, ou, sur une liaison établie, dans une primitive P-ALTER-CONTEXT) une liste de nouvelles syntaxes abstraites (pouvant inclure des syntaxes abstraites de caractères) ou d'éliminer certaines syntaxes abstraites, et permettent au répondeur d'effectuer son choix dans cette liste.

G.6.6 L'initiateur de la connexion peut exprimer ses préférences par l'ordre des syntaxes abstraites dans la liste, ou en utilisant le paramètre données d'utilisateur qui peut être utilisé par le concepteur de l'application pour expliquer pourquoi l'utilisation de cette nouvelle syntaxe abstraite est proposée. Il peut indiquer par exemple que toutes les syntaxes abstraites (de caractères) sont proposées pour être utilisées dans un même but donné, ou que son intention est de permettre le choix d'une seule de ces syntaxes à des fins multiples.

G.6.7 Des syntaxes abstraites de caractères (et les syntaxes de transfert correspondantes) sont définies dans plusieurs Recommandations UIT-T et Normes internationales; des syntaxes abstraites (et/ou des syntaxes de transfert) de caractères supplémentaires peuvent également être définies par tout organisme habilité à affecter des identificateurs d'objets.

G.6.8 Dans ISO/CEI 10646-1, une syntaxe abstraite de caractères est définie (et des identificateurs d'objets affectés) pour la collection complète de caractères, pour chacune des collections de caractères définies comme sous-ensemble de la collection complète (BASIC LATIN, BASIC SYMBOLS, etc.) et pour chaque combinaison possible des collections de caractères définies. Deux syntaxes de transfert de caractères sont aussi définies pour identifier les différentes options (en particulier les représentations sur 16 bits et sur 32 bits) dans ISO/CEI 10646-1.

Annexe H

Caractéristiques remplacées

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

Un certain nombre de caractéristiques figurant dans les versions antérieures de la présente Recommandation | Norme internationale (à savoir la Rec. X.208 du CCITT (1988) | ISO/CEI 8824:1990) ont été remplacées et ne font plus partie de la notation ASN.1. Elles peuvent toutefois se trouver dans certains modules ASN.1 existants. La présente annexe décrit ces caractéristiques et explique comment obtenir les fonctions avec les notations qui les remplacent.

H.1 Utilisation des identificateurs devenus obligatoires

Un type nommé "NamedType" et une valeur nommée "NamedValue" étaient notés auparavant:

NamedType ::= identifieur Type | Type | SelectionType
NamedValue ::= identifieur Value | Value

Ces notations sont remplacées par:

NamedType ::= identifieur Type
NamedValue ::= identifieur Value

car les premières pouvaient conduire à des ambiguïtés syntaxiques.

Des identificateurs peuvent être ajoutés aux notations de types nommés "NamedType" dans les anciennes spécifications ASN.1 sans affecter le codage du type (bien qu'il soit nécessaire ensuite d'apporter des modifications aux notations ASN.1 pour toute utilisation de notation de valeur correspondante). Une telle modification doit être apportée au titre d'un rapport de défaut ou dans le cadre d'une révision de la Recommandation | Norme internationale concernée.

H.2 Valeur du type choix

Une valeur de type choix était notée auparavant:

ChoiceValue ::= NamedValue
NamedValue ::= identifieur Value | Value

Cette notation est remplacée par:

ChoiceValue ::= identifieur ":" Value

car la première pouvait conduire à des ambiguïtés syntaxiques.

H.3 Type ANY

Le type ANY (quelconque) était défini dans les versions antérieures de la présente Recommandation | Norme internationale. A titre d'information, la partie de la version précédente de la présente Recommandation | Norme internationale dans laquelle cette capacité était spécifiée est reproduite à l'Annexe I.

Le type ANY était normalement utilisé pour laisser "en blanc" un élément d'une spécification, à charge de le remplir dans une autre spécification. La notation "AnyType", autorisée comme forme possible de "Type", était spécifiée comme suit:

AnyType ::= ANY | ANY DEFINED BY identifieur

Une valeur de type ANY était notée auparavant:

AnyValue ::= Type Value

et cette notation avait été ultérieurement modifiée en:

AnyValue ::= Type : Value

car la première formulation posait des problèmes de traitement machine de la notation ASN.1.

Il était vivement recommandé d'utiliser la seconde forme de cette notation. Dans cette forme, qui n'était autorisée que lorsque le type ANY était l'un des types composant un type ensemble ou séquence, une autre composante de l'ensemble ou de la séquence (celle dont l'identificateur était désigné) indiquait, par sa valeur d'entier ou d'identificateur d'objet (ou un choix de ces éléments), le type effectif gouvernant la composante ANY. La correspondance entre ces valeurs et des types ASN.1 donnés pouvait être considérée comme un "tableau" faisant partie de la syntaxe abstraite. En l'absence de l'identificateur déclaré par DEFINED BY, (première forme de la notation de type), la notation ne comportait aucune indication quant à la manière de déterminer le type de champ. Cela aboutissait fréquemment à des spécifications dans lesquelles le "blanc" persistait même à l'étape de mise en œuvre.

Le type ANY est maintenant remplacé par la possibilité de spécifier des classes d'objets informationnels, puis de se référer aux champs de ces classes depuis l'intérieur des définitions de types (voir la Rec. UIT-T X.681 | ISO/CEI 8824-2). Comme les champs peuvent être définis de manière à accueillir un type ASN.1 arbitraire, la possibilité de laisser des éléments "en blanc" est fondamentalement conservée. Cependant, cette nouvelle caractéristique permet également de spécifier une "contrainte tabulaire" dans laquelle un ensemble d'objets informationnels donné (un ensemble d'objets de la classe considérée) est explicitement cité comme contraignant le type. Cette dernière capacité englobe celle qu'offrait la notation "ANY DEFINED BY identificateur".

De plus, des utilisations prédéfinies de ces nouvelles capacités sont établies et correspondent à diverses utilisations courantes du type ANY (voir la Rec. UIT-T X.681 | ISO/CEI 8824-2). Par exemple, une séquence contenant un identificateur d'objet et un type ANY, souvent utilisée auparavant pour acheminer une valeur arbitraire en même temps qu'une indication de son type, peut maintenant être décrite par:

INSTANCE OF MACHINTRUC

où MACHINTRUC se définit ainsi:

MACHINTRUC ::= TYPE-IDENTIFIER

Cette notation entraîne le remplacement de "INSTANCE OF MACHINTRUC" par un identificateur d'objet de la classe MACHINTRUC avec le type identifié par cet identificateur. Un exemple est donné d'une telle utilisation au F.2.18.

Des couples particuliers d'identificateurs d'objets et de types sont définis comme les objets informationnels de la classe **MACHINTRUC** et, si nécessaire, des ensembles particuliers de ceux-ci peuvent être définis et utilisés pour contraindre la structure **INSTANCE OF** de façon que seuls ces objets de l'ensemble puissent apparaître.

La capacité de macro-notation était souvent utilisée comme un moyen semi-formel de définir des tableaux d'objets informationnels régissant l'utilisation d'un type ANY associé. Elle est aussi remplacée par les nouvelles notations.

H.4 Capacité de macro-notation

La capacité de macro-notation permettait à l'utilisateur de la notation ASN.1 d'étendre cette notation par des macro-notations. A titre d'information, la partie de la version précédente de la présente Recommandation | Norme internationale dans laquelle cette capacité était spécifiée est reproduite à l'Annexe J.

La capacité de macro-notation était essentiellement utilisée pour définir une notation spécifiant des objets informationnels. Elle est maintenant directement incorporée dans la notation ASN.1 (voir la Rec. UIT-T X.681 | ISO/CEI 8824-2) et n'a plus besoin du caractère très général (et des risques correspondants) d'une notation définie par l'utilisateur.

En outre, la seule autre utilisation des macro-notations semble être de définir des expressions devant être complétées par des paramètres quelconques avant de devenir des types ASN.1 complètement définis. Cette opération est maintenant assurée par la capacité plus générale de paramétrage (voir la Rec. UIT-T X.683 | ISO/CEI 8824-4).

Annexe I

Notation du type ANY (quelconque)

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

La présente annexe est donnée à titre historique et la notation qui y est décrite ne fait plus partie de l'ASN.1.

I.1 Notation du type ANY

I.1.1 Un type ANY (quelconque) est produit par la notation "AnyType":

AnyType ::= ANY | ANY DEFINED BY identificateur

NOTE – L'utilisation d'un type ANY dans une Recommandation ou une Norme ISO génère une spécification qui reste incomplète sans une spécification additionnelle. La structure "ANY DEFINED BY" offre la possibilité de spécifier dans une instance de communication le type destiné au champ ANY et un pointeur qui désigne sa sémantique. Si les règles d'application suivantes sont respectées, ANY peut fournir une spécification complète. L'utilisation d'un type ANY sans la structure "DEFINED BY" est déconseillée.

I.1.2 La forme "DEFINED BY" ne sera utilisée que si le type ANY ou le type qui en dérive est l'un des types composants d'un type séquence ou ensemble (le type contenant).

I.1.3 L'identificateur de la forme "DEFINED BY" doit également apparaître dans un type nommé "NamedType" qui spécifie un autre composant, non optionnel, du type contenant. Le "Type" de la notation "NamedType" doit dériver d'un type entier, d'un type identificateur d'objet ou d'un choix formé de ces deux types.

I.1.4 Quand le "Type" de la notation "NamedType" est du type entier, le document employant la déclaration "DEFINED BY" comportera ou fera explicitement référence à une liste unique spécifiant le type ASN.1 à véhiculer par le champ ANY pour chaque valeur autorisée du type entier. Il y aura une et une seule liste couvrant toutes les instances de communication du type contenant.

I.1.5 Quand le "Type" de la notation "NamedType" est du type identificateur d'objet, il faut que des registres associent à chaque valeur d'identificateur d'objet attribuée, un type ASN.1 unique (pouvant être un type choix) à véhiculer par le champ ANY.

NOTES

1 Il peut exister un nombre arbitraire de registres à cet effet associant une valeur d'identificateur d'objet à un type ASN.1.

2 L'enregistrement des valeurs, effectué à des fins d'interconnexion de systèmes ouverts, devrait se faire dans les Recommandations et les Normes ISO utilisant la notation. Quand il est prévu qu'une instance de notation "ANY DEFINED BY" sera enregistrée par une autre Autorité d'enregistrement internationale, ceci doit être indiqué dans le document utilisant la notation.

3 La principale différence entre l'utilisation d'identificateurs de type entier et de type identificateur d'objet, est que l'identificateur entier pointe une liste unique, contenue dans la norme qui l'utilise, alors que l'identificateur d'objet permet de pointer un ensemble non clos de types déterminés par toute autorité habilitée à affecter des identificateurs d'objets.

I.1.6 Ce type possède une étiquette indéterminée et ne doit pas être utilisé lorsque la présente Recommandation | Norme internationale exige des étiquettes distinctes (voir 22.5, 24.3, 26.2 et 26.4).

I.1.7 Une valeur de type ANY est déclarée par la notation ASN.1 "AnyValue":

AnyValue ::= Type : Value

où "Type" est la notation du type choisi et "Value" une notation valide d'une valeur de ce type.

Annexe J

Les macro-notations

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

La présente annexe est donnée à titre historique et la notation qui y est décrite ne fait plus partie de l'ASN.1.

J.1 Introduction

La notation ASN.1 comporte un mécanisme permettant à son utilisateur de définir une nouvelle notation avec laquelle il peut construire et désigner des types ASN.1 ou spécifier des valeurs de ces types. La nouvelle notation se définit en utilisant la notation "MacroDefinition". Une telle macro-définition spécifie simultanément une nouvelle notation de structuration et de référencement de type et une nouvelle notation de spécification de valeur.

Avec une macro-définition, l'utilisateur ASN.1 spécifie la nouvelle notation au moyen d'un ensemble de productions, d'une façon similaire à celle de la présente Recommandation | Norme internationale. L'auteur de la macro-définition:

- a) spécifie la syntaxe complète à utiliser pour définir tous les types pris en charge par la macro-définition; (cette spécification de syntaxe est invoquée à des fins d'analyse syntaxique par chaque occurrence du nom de la macro-notation dans la notation de type ASN.1);
- b) spécifie la syntaxe complète à utiliser pour une valeur d'un de ces types; (cette spécification de syntaxe est invoquée à des fins d'analyse syntaxique chaque fois qu'une valeur du type de la macro-définition est prévue);
- c) spécifie, comme valeur d'un type standard ASN.1 (d'une complexité arbitraire), le type et la valeur résultants pour toutes les instances de la macro-notation de valeur.

Une instance de la syntaxe définie par la macro-définition peut contenir des instances de types ou de valeurs (en notation standard ASN.1). Ces types ou valeurs (qui apparaissent dans l'utilisation de la macro-notation) peuvent être associés, pour la durée de l'analyse syntaxique, à une **référence locale de type** ou **de valeur**, par des déclarations appropriées dans la macro-définition. Il est également possible d'encapsuler dans la macro-définition des affectations de type standard ASN.1. Ces affectations sont activées lorsque la catégorie syntaxique associée est appliquée à un ou plusieurs items de l'instance de la nouvelle notation en cours d'analyse. Leur durée de vie est limitée à celle de l'analyse.

Lorsqu'une valeur de la nouvelle notation est analysée, les affectations faites pendant l'analyse de la notation du type correspondant sont accessibles. Une telle analyse est considérée comme précédant logiquement l'analyse de chaque instance de la notation de valeur.

Le type et la valeur résultant d'une instance d'utilisation de la nouvelle notation de valeur sont déterminés par la valeur (et le type de cette valeur) qui sont affectés en fin de compte à la référence locale distinguée de la valeur identifiée par le mot clé VALUE, selon le traitement de la macro-définition pour la nouvelle notation de type, suivie par celle de la nouvelle notation de valeur.

Chaque macro-définition définit une notation (syntaxe) de définition de type, et une notation (syntaxe) de définition de valeur. Le type ASN.1 qui est défini par une instance de la nouvelle notation de type peut, mais pas nécessairement, dépendre de l'instance de la notation de valeur à laquelle le type est associé. De ce point de vue, l'utilisation de la nouvelle notation de type est similaire à un choix: l'étiquette est indéterminée. Par conséquent, la nouvelle notation ne peut être utilisée lorsqu'il est spécifié de déclarer une étiquette connue; il n'est pas possible non plus d'y déclarer un étiquetage implicite.

J.2 Extensions aux items et au jeu de caractères ASN.1

Les caractères | et > sont utilisés en macro-notation.

Les items spécifiés dans les points suivants sont également utilisés.

J.2.1 Macro-référence

Nom de l'item – macroreference

Une macro-référence se compose de la séquence de caractères spécifiée pour une référence de type "typereference" au 9.2, sauf que tous les caractères seront des majuscules. Dans un module donné, une même séquence de caractères ne doit pas être utilisée à la fois pour une référence de type et une macro-référence.

J.2.2 Référence de production

Nom de l'item – productionreference

Une référence "productionreference" se compose de la séquence de caractères spécifiée pour une référence de type "typereference" au 9.2.

J.2.3 Référence locale de type

Nom de l'item – localtypereference

Une référence "localtypereference" se compose de la séquence de caractères spécifiée pour une référence de type "typereference" au 9.2. La référence "localtypereference" est utilisée comme identificateur pour les types qui sont reconnus au cours de l'analyse syntaxique d'une instance de la nouvelle notation de type ou de valeur.

J.2.4 Référence locale de valeur

Nom de l'item – localvaluereference

Une référence "localvaluereference" est composée de la séquence de caractères spécifiée pour une référence de type "typereference" au 9.2. Une référence "localvaluereference" est utilisée comme identificateur pour les valeurs qui sont reconnues au cours de l'analyse syntaxique d'une instance de la nouvelle notation de type ou de valeur.

NOTE – Une référence locale de valeur commence par une majuscule.

J.2.5 Item de choix exclusif

Nom de l'item – "|"

Cet item se compose du seul caractère |.

J.2.6 Item de clôture de définition

Nom de l'item – ">"

Cet item se compose du seul caractère >.

NOTE – L'item < de début de définition est défini au 9.15.

J.2.7 Item terminal syntaxique

Nom de l'item – astring

Un item "astring" se compose d'un nombre arbitraire (éventuellement zéro) de caractères du jeu de caractères ASN.1 (voir l'article 8) placés entre guillemets (""). S'il fait partie de l'item "astring", le guillemet sera représenté par un double guillemet ("").

NOTE – L'emploi de l'item "astring" en macro-notation indique l'occurrence au point correspondant de la syntaxe analysée, des caractères placés entre les guillemets.

J.2.8 Items mots clés de catégorie syntaxique

Nom des items –

"string"	(chaîne)
"identifieur"	(identificateur)
"number"	(numéro)
"empty"	(vide)

Les items ci-dessus apparaîtront dans les macro-notations sous la forme de la séquence de caractères sans les guillemets qui les encadrent. Ces items sont utilisés en macro-notation pour spécifier l'occurrence, dans une instance de la nouvelle notation, de certaines séquences de caractères. Le Tableau J.1 indique les séquences de caractères de la nouvelle notation spécifiées par chaque item, et signale l'article de la présente Recommandation | Norme internationale qui définit cette séquence de caractères.

NOTE – Pour des raisons historiques, la macro-notation ne différencie pas les identificateurs et les références par le caractère majuscule ou minuscule de la première lettre.

Tableau J.1 – Séquences spécifiées par les items

Nom de l'item	Définition
"string"	Toute séquence de caractères
"identifiant"	9.3 – Identificateurs
"number"	9.8 – Nombres
"empty"	9.7 – Vide

J.2.9 Autres items mots clés

Nom des items –

MACRO
 TYPE
 NOTATION
 VALUE
 value
 type

Les items ci-dessus apparaîtront dans les macro-notations sous la forme de la seule séquence de caractères.

Les items spécifiés dans J.2.2 à J.2.4 inclus ne seront pas l'une des séquences du J.2.9, sauf lorsqu'ils sont utilisés comme spécifié ci-dessous.

Le mot clé "MACRO" sera utilisé pour introduire une macro-définition. Le mot clé "TYPE NOTATION" sera utilisé comme nom de la production qui définit la nouvelle notation de type. Le mot clé "VALUE NOTATION" sera utilisé comme nom de la production qui définit la nouvelle notation de valeur. Le mot clé "VALUE" sera utilisé comme la référence "localvaluereference" à laquelle la valeur résultante est affectée. Le mot clé "value" sera utilisé pour spécifier qu'une instance de la nouvelle notation contient, à ce point, en notation ASN.1 standard, une certaine valeur d'un type donné (spécifié dans la macro-définition). Le mot clé "type" sera utilisé pour spécifier qu'une instance de la nouvelle notation contient, à ce point, en notation ASN.1 standard, un certain "Type".

J.3 Notation de macro-définition

J.3.1 Une macro-notation sera définie par la notation "MacroDefinition":

```

MacroDefinition ::=
    macroreference
    MACRO
    "::="
    MacroSubstance

MacroSubstance ::=
    BEGIN MacroBody END |
    macroreference      |
    Externalmacroreference

MacroBody ::=
    TypeProduction
    ValueProduction
    SupportingProductions

TypeProduction ::=
    TYPE NOTATION
    "::="
    MacroAlternativeList

ValueProduction ::=
    VALUE NOTATION
    "::="
    MacroAlternativeList
    
```

SupportingProductions ::=

ProductionList |
empty

ProductionList ::=

Production |
ProductionList Production

Production ::=

productionreference
"::=" |
MacroAlternativeList

Externalmacroreference ::=

modulereference "." **macroreference**

J.3.2 Si la forme "macroreference" de la notation "MacroSubstance" est choisie, le module contenant la macro-définition doit:

- a) soit contenir une autre macro-définition définissant cette "macroreference";
- b) soit contenir la référence "macroreference" parmi ses symboles importés "SymbolsImported".

J.3.3 Si la forme "Externalmacroreference" de la notation "MacroSubstance" est choisie, le module désigné par "modulereference" doit contenir une macro-définition définissant la référence "macroreference". La définition associée est alors également associée à la référence "macroreference" ainsi définie.

J.3.4 La chaîne de définitions qui peut résulter de l'application répétée des règles de J.3.2 et J.3.3 se terminera par une définition "MacroDefinition" qui utilise la forme "BEGIN MacroBody END".

J.3.5 Chaque référence "productionreference" apparaissant dans une définition de symbole "SymbolDefn" (voir J.3.9) apparaîtra une et une seule fois comme premier élément d'une "Production".

J.3.6 Chaque instance de la nouvelle notation de type commencera par la séquence de caractères de "macroreference" suivie de l'une des séquences de caractères désignées par la déclaration "TYPE NOTATION" après application des productions spécifiées dans la macro-définition.

J.3.7 Chaque instance de la nouvelle notation de valeur sera composée d'une des séquences de caractères désignées par la déclaration "VALUE NOTATION", après application des productions spécifiées dans la macro-définition.

J.3.8 La liste des macro-notations possibles "MacroAlternativeList" d'une production spécifie les ensembles possibles de séquences de caractères données en référence par cette production. Elle est spécifiée par:

MacroAlternativeList ::=

MacroAlternative |
MacroAlternativeList "|" **MacroAlternative**

L'ensemble des séquences de caractères indiquées par la liste "MacroAlternativeList" est composé de toutes les séquences de caractères qui sont citées en référence par les différentes productions "MacroAlternative" de la liste "MacroAlternativeList".

J.3.9 La notation d'une forme "MacroAlternative" est:

MacroAlternative ::= SymbolList

SymbolList ::=

SymbolElement |
SymbolList SymbolElement

SymbolElement ::=

SymbolDefn |
EmbeddedDefinitions

SymbolDefn ::=

astring |
productionreference |
"string" |
"identifiant" |
"number" |
"empty" |

```

type |
type(localtypereference) |
value(MacroType) |
value(localvaluereference MacroType) |
value(VALUE MacroType)

```

```

MacroType ::=
localtypereference |
Type

```

NOTE – Dans une macro-notation, une notation "MacroType" qui y est définie peut figurer partout où la notation ASN.1 peut spécifier un "Type".

Une forme "MacroAlternative" désigne toutes les chaînes de caractères formées en prenant n'importe laquelle des chaînes désignées par la première notation "SymbolDefn" de la liste "SymbolList", suivie de n'importe laquelle des chaînes désignées par la deuxième notation "SymbolDefn" de la liste "SymbolList", et ainsi de suite jusques et y compris la dernière notation "SymbolDefn" de la liste.

NOTE – Les définitions encapsulées "EmbeddedDefinitions" (s'il y en a) ne jouent aucun rôle direct dans la détermination de ces chaînes.

J.3.10 Une chaîne "astring" désigne la séquence de caractères sans les guillemets qui l'encadrent.

J.3.11 Une référence "productionreference" désigne toute séquence de caractères spécifiée par la "Production" qu'elle identifie.

J.3.12 Les séquences de caractères désignées par les quatre formes possibles suivantes de la notation "SymbolDefn" sont spécifiées dans le Tableau J.1.

NOTE – Les séquences de caractères désignées par la chaîne "string" doivent se terminer dans une instance de la macro-notation, par l'apparition d'une séquence référencée par la définition "SymbolDefn" suivante de la liste "SymbolList".

J.3.13 Un "type" désigne toute séquence de symboles qui forme une notation de "Type" comme le spécifie 14.1.

NOTE – Le type "ReferencedType" du 14.1 peut dans ce cas contenir une référence "localtypereference" désignant un type défini dans la macro-notation.

J.3.14 Une notation "type(localtypereference)" désigne toute séquence de symboles formant un "Type" comme le spécifie 14.1, mais affecte en plus ce type à la référence "localtypereference". Cette même référence "localtypereference" peut faire l'objet d'une affectation ultérieure.

J.3.15 Une notation "value(MacroType)" désigne toute séquence de symboles formant une notation "Value" (comme le spécifie 14.7) pour le type spécifié par "MacroType".

J.3.16 Une notation "value(localvaluereference MacroType)" désigne toute séquence de symboles formant une notation "Value" (comme le spécifie 14.7) pour le type spécifié par "MacroType", mais affecte en plus la valeur spécifiée par la notation de valeur à la référence "localvaluereference". Cette référence "localvaluereference" peut faire l'objet d'une affectation ultérieure.

J.3.17 Une notation "value(VALUE MacroType)" désigne toute séquence de symboles formant une notation "Value" (comme le spécifie 14.7) pour le type spécifié par "MacroType", mais renvoie en plus la valeur en tant que valeur spécifiée par la notation de valeur. Le type de la valeur renvoyée est le type désigné par "MacroType".

J.3.18 Lors de l'analyse de toute instance correcte de la nouvelle notation de valeur, VALUE reçoit une et une seule affectation (comme spécifié au J.3.17 ou au J.3.19).

J.3.19 La notation de définitions encapsulées "EmbeddedDefinitions" est:

```

EmbeddedDefinitions ::= "<" EmbeddedDefinitionList ">"

```

```

EmbeddedDefinitionList ::=
EmbeddedDefinition |
EmbeddedDefinitionList
EmbeddedDefinition

```

```

EmbeddedDefinition ::=
LocalTypeassignment |
LocalValueassignment

```

```

LocalTypeassignment ::=
localtypereference
"::="
MacroType

```

LocalValueassignment ::=
localvaluereference
MacroType
::="
MacroValue

MacroValue ::=
Value |
localvaluereference

L'affectation d'un type "MacroType" à une référence "localtypereference" (ou d'une valeur "MacroValue" à une référence "localvaluereference") dans une liste "EmbeddedDefinitions" prend effet au cours de l'analyse syntaxique d'une instance de la nouvelle notation au moment où la notation "EmbeddedDefinitions" est rencontrée, et reste en vigueur jusqu'à la redéfinition de la référence "localtypereference" ou "localvaluereference".

NOTES

1 L'utilisation de la référence associée "localtypereference" ou "localvaluereference" ailleurs dans la notation "MacroAlternative", implique des hypothèses sur la nature de l'algorithme d'analyse syntaxique. Ces hypothèses devraient être précisées par un commentaire. Par exemple, l'utilisation de la référence "localtypereference" textuellement à la suite de la notation "EmbeddedDefinitions" implique une analyse syntaxique de gauche à droite.

2 La référence "localvaluereference" "VALUE" peut se voir affecter une valeur soit par la production "value(VALUE MacroType)" soit par une définition "EmbeddedDefinition". Dans les deux cas, la valeur est renvoyée comme le spécifie J.3.17.

J.4 Utilisation de la nouvelle notation

Chaque fois qu'une notation "Type" (ou "Value") est appelée par la présente Recommandation | Norme internationale, une instance de la notation de type (ou de la notation de valeur) définie par une macro-notation dans le même module peut être utilisée, à condition que cette macro-notation soit:

- a) définie dans le même module; ou
- b) importée dans le module du fait de la présence de la référence "macroreference" dans la déclaration "SymbolsImported" du module.

Pour rendre cette dernière forme possible, une référence "macroreference" peut apparaître comme un symbole "Symbol" du 10.1.

NOTES

1 Cette extension à la notation standard ASN.1 n'apparaît pas dans le corps de la présente Recommandation | Norme internationale.

2 Il est possible de construire des modules comprenant des séquences d'affectation de type et des macro-définitions, quelle que soit la complexité résultante de l'analyse syntaxique des valeurs par défaut DEFAULT.

Annexe K

Récapitulatif de la notation ASN.1

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

Les items ci-dessous sont définis dans l'article 9:

typereference	BEGIN	ISO646String
identifiant	BIT	MAX
valuereference	BMPString	MIN
modulereference	BOOLEAN	MINUS-INFINITY
comment	BY	NULL
empty	CHARACTER	NumericString
number	CHOICE	OBJECT
bstring	CLASS	ObjectDescriptor
hstring	COMPONENT	OCTET
cstring	COMPONENTS	OF
"::="	CONSTRAINED	OPTIONAL
".."	DEFAULT	PDV
"..."	DEFINITIONS	PLUS-INFINITY
"{"	EMBEDDED	PRESENT
"}"	END	PrintableString
"<"	ENUMERATED	PRIVATE
","	EXCEPT	REAL
."	EXPLICIT	SEQUENCE
"("	EXPORTS	SET
")"	EXTERNAL	SIZE
"["	FALSE	STRING
"]"	FROM	SYNTAX
"_"	GeneralizedTime	T61String
":"	GeneralString	TAGS
","	GraphicString	TeletexString
"@"	IA5String	TRUE
" "	TYPE-IDENTIFIER	UNION
"!"	IDENTIFIER	UNIQUE
"^"	IMPLICIT	UNIVERSAL
ABSENT	IMPORTS	UniversalString
ABSTRACT-SYNTAX	INCLUDES	UTCTime
ALL	INSTANCE	VideotexString
APPLICATION	INTEGER	VisibleString
AUTOMATIC	INTERSECTION	WITH

Les productions suivantes sont utilisées dans la présente Recommandation | Norme internationale, avec les items ci-dessus comme symboles terminaux:

```

ModuleDefinition ::= ModuleIdentifie
DEFINITIONS
TagDefault
"::="
BEGIN
ModuleBody
END

ModuleIdentifie ::= modulereference
DefinitiveIdentifie

DefinitiveIdentifie ::= "{" DefinitiveObjIdComponentList "}" |
empty

DefinitiveObjIdComponentList ::=
DefinitiveObjIdComponent |
DefinitiveObjIdComponent DefinitiveObjIdComponentList

DefinitiveObjIdComponent ::=
NameForm |
DefinitiveNumberForm |
DefinitiveNameAndNumberForm

DefinitiveNumberForm ::= number

DefinitiveNameAndNumberForm ::= identifie "(" DefinitiveNumberForm ")"

TagDefault ::= EXPLICIT TAGS |
IMPLICIT TAGS |
AUTOMATIC TAGS |
empty

ModuleBody ::= Exports Imports AssignmentList |
empty

Exports ::= EXPORTS SymbolsExported ";" |
empty

SymbolsExported ::= SymbolList |
empty

Imports ::= IMPORTS SymbolsImported ";" |
empty

SymbolsImported ::= SymbolsFromModuleList |
empty

SymbolsFromModuleList ::=
SymbolsFromModule |
SymbolsFromModuleList SymbolsFromModule

SymbolsFromModule ::= SymbolList FROM GlobalModuleReference

GlobalModuleReference ::= modulereference AssignedIdentifie

AssignedIdentifie ::= ObjectIdentifieValue |
DefinedValue |
empty

SymbolList ::= Symbol | Symbol "," SymbolList

Symbol ::= Reference | ParameterizedReference

Reference ::=
typerreference |
valuereference |
objectclassreference |
objectreference |
objectsetreference

AssignmentList ::= Assignment | AssignmentList Assignment

```

```

Assignment ::=
    TypeAssignment |
    ValueAssignment |
    ValueSetTypeAssignment |
    ObjectClassAssignment |
    ObjectAssignment |
    ObjectSetAssignment |
    ParameterizedAssignment

Externaltypereference ::=
    modulereference
    "."
    typereference

Externalvaluereference ::=
    modulereference
    "."
    valuereference

DefinedType ::=
    Externaltypereference |
    typereference |
    ParameterizedType |
    ParameterizedValueSetType

DefinedValue ::=
    Externalvaluereference |
    valuereference |
    ParameterizedValue

AbsoluteReference ::= "@" GlobalModuleReference
    "."
    ItemSpec

ItemSpec ::=
    typereference |
    ItemId "." ComponentId

ItemId ::= ItemSpec

ComponentId ::=
    identifier | number | "*"

TypeAssignment ::= typereference
    "::~="
    Type

ValueAssignment ::= valuereference
    Type
    "::~="
    Value

ValueSetTypeAssignment ::= typereference
    Type
    "::~="
    ValueSet

ValueSet ::= "{" ElementSetSpec "}"

Type ::= BuiltinType | ReferencedType | ConstrainedType

BuiltinType ::=
    BitStringType |
    BooleanType |
    CharacterStringType |
    ChoiceType |
    EmbeddedPDVType |
    EnumeratedType |
    ExternalType |
    InstanceOfType |
    IntegerType |
    NullType |

```

ObjectClassFieldType |
 ObjectIdentifierType |
 OctetStringType |
 RealType |
 SequenceType |
 SequenceOfType |
 SetType |
 SetOfType |
 TaggedType

NamedType ::= identifier Type | SelectionType

ReferencedType ::=
 DefinedType |
 UsefulType |
 SelectionType |
 TypeFromObject |
 ValueSetFromObjects

Value ::= BuiltinValue | ReferencedValue

BuiltinValue ::=
 BitStringValue |
 BooleanValue |
 CharacterStringValue |
 ChoiceValue |
 EmbeddedPDVValue |
 EnumeratedValue |
 ExternalValue |
 InstanceOfValue |
 IntegerValue |
 NullValue |
 ObjectClassFieldValue |
 ObjectIdentifierValue |
 OctetStringValue |
 RealValue |
 SequenceValue |
 SequenceOfValue |
 SetValue |
 SetOfValue |
 TaggedValue

ReferencedValue ::=
 DefinedValue |
 ValueFromObject

NamedValue ::= identifier Value

BooleanType ::= BOOLEAN

BooleanValue ::= TRUE | FALSE

IntegerType ::=
 INTEGER |
 INTEGER "{" NamedNumberList "}"

NamedNumberList ::=
 NamedNumber |
 NamedNumberList "," NamedNumber

NamedNumber ::=
 identifier "(" SignedNumber ")" |
 identifier "(" DefinedValue ")"

SignedNumber ::= number | "-" number

IntegerValue ::= SignedNumber | identifier

EnumeratedType ::=
 ENUMERATED "{" Enumeration "}"

Enumeration ::=
 EnumerationItem | EnumerationItem "," Enumeration

EnumerationItem ::=
 identifieur | NamedNumber

EnumeratedValue ::=
 identifieur

RealType ::= REAL

RealValue ::=
 NumericRealValue | SpecialRealValue

NumericRealValue ::= 0 |
 SequenceValue -- Value of the associated sequence type

SpecialRealValue ::=
 PLUS-INFINITY | MINUS-INFINITY

BitStringType ::= BIT STRING | BIT STRING "{" NamedBitList "}"

NamedBitList ::= NamedBit | NamedBitList "," NamedBit

NamedBit ::= identifieur "(" number ")" |
 identifieur "(" DefinedValue ")"

BitStringValue ::= bstring | hstring | "{" IdentifierList "}" | "{" "}"

IdentifierList ::= identifieur | IdentifierList "," identifieur

OctetStringType ::= OCTET STRING

OctetStringValue ::= bstring | hstring

NullType ::= NULL

NullValue ::= NULL

SequenceType ::= SEQUENCE "{" ComponentTypeList "}" |
 SEQUENCE "{" "}"

ComponentTypeList ::= ComponentType |
 ComponentTypeList "," ComponentType

ComponentType ::= NamedType |
 NamedType OPTIONAL |
 NamedType DEFAULT Value |
 COMPONENTS OF Type

SequenceValue ::= "{" ComponentValueList "}" | "{" "}"

ComponentValueList ::= NamedValue |
 ComponentValueList "," NamedValue

SequenceOfType ::= SEQUENCE OF Type

SequenceOfValue ::= "{" ValueList "}" | "{" "}"

ValueList ::= Value | ValueList "," Value

SetType ::= SET "{" ComponentTypeList "}" | SET "{" "}"

SetValue ::= "{" ComponentValueList "}" | "{" "}"

SetOfType ::= SET OF Type

SetOfValue ::= "{" ValueList "}" | "{" "}"

ChoiceType ::= CHOICE "{" AlternativeTypeList "}"

AlternativeTypeList ::= NamedType |
 AlternativeTypeList "," NamedType

ChoiceValue ::= identifieur ":" Value

SelectionType ::= identifieur "<" Type

TaggedType ::= Tag Type |
 Tag IMPLICIT Type |
 Tag EXPLICIT Type

Tag ::= "[" Class ClassNumber "]"
ClassNumber ::= number | DefinedValue
Class ::= UNIVERSAL |
APPLICATION |
PRIVATE |
empty
TaggedValue ::= Value
EmbeddedPDVType ::= EMBEDDED PDV
EmbeddedPDVValue ::= SequenceValue
ExternalType ::= EXTERNAL
ExternalValue ::= SequenceValue
ObjectIdentifierType ::= OBJECT IDENTIFIER
ObjectIdentifierValue ::= "{" ObjIdComponentList "}" |
"{" DefinedValue ObjIdComponentList "}"
ObjIdComponentList ::= ObjIdComponent |
ObjIdComponent ObjIdComponentList
ObjIdComponent ::= NameForm |
NumberForm |
NameAndNumberForm
NameForm ::= identifier
NumberForm ::= number | DefinedValue
NameAndNumberForm ::= identifier "(" NumberForm ")"
CharacterStringType ::= RestrictedCharacterStringType | UnrestrictedCharacterStringType
RestrictedCharacterStringType ::= BMPString |
GeneralString |
GraphicString |
IA5String |
ISO646String |
NumericString |
PrintableString |
TeletexString |
T61String |
UniversalString |
VideotexString |
VisibleString
RestrictedCharacterStringValue ::= cstring | CharacterStringList | Quadruple | Tuple
CharacterStringList ::= "{" CharSyms "}"
CharSyms ::= CharsDefn | CharSyms "," CharsDefn
CharsDefn ::= cstring | DefinedValue
Quadruple ::= "{" Group "," Plane "," Row "," Cell "}"
Group ::= number
Plane ::= number
Row ::= number
Cell ::= number
Tuple ::= "{" TableColumn "," TableRow "}"
TableColumn ::= number
TableRow ::= number
UnrestrictedCharacterStringType ::= CHARACTER STRING
CharacterStringValue ::= RestrictedCharacterStringValue | UnrestrictedCharacterStringValue
UnrestrictedCharacterStringValue ::= SequenceValue
UsefulType ::= typereference

Les types de chaîne de caractères ci-dessous sont définis au 34.1:

NumericString VisibleString
PrintableString ISO646String
TeletexString IA5String
T61String GraphicString
VideotexString GeneralString
UniversalString BMPString

Les types utiles ci-dessous sont définis dans les articles 39 à 41:

GeneralizedTime
UTCTime
ObjectDescriptor

Les productions ci-dessous sont utilisées dans les articles 42 à 45:

ConstrainedType ::=
 Type Constraint |
 TypeWithConstraint

TypeWithConstraint ::=
 SET Constraint OF Type |
 SET SizeConstraint OF Type |
 SEQUENCE Constraint OF Type |
 SEQUENCE SizeConstraint OF Type

Constraint ::= "(" ConstraintSpec ExceptionSpec ")"

ConstraintSpec ::=
 SubtypeConstraint |
 GeneralConstraint

ExceptionSpec ::= "!" ExceptionIdentification | empty

ExceptionIdentification ::= SignedNumber |
 DefinedValue |
 Type ":" Value

SubtypeConstraint ::= ElementSetSpec

ElementSetSpec ::= Unions | ALL Exclusions

Unions ::= Intersections |
 UElems UnionMark Intersections

UElems ::= Unions

Intersections ::= IntersectionElements |
 IElems IntersectionMark IntersectionElements

IElems ::= Intersections

IntersectionElements ::= Elements | Elems Exclusions

Elems ::= Elements

Exclusions ::= EXCEPT Elements

UnionMark ::= "|" | UNION

IntersectionMark ::= "^" | INTERSECTION

Elements ::=
 SubtypeElements |
 ObjectSetElements |
 "(" ElementSetSpec ")"

SubtypeElements ::=
 SingleValue |
 ContainedSubtype |
 ValueRange |
 PermittedAlphabet |
 SizeConstraint |
 TypeConstraint |
 InnerTypeConstraints

SingleValue ::= Value

ContainedSubtype ::= Includes Type

Includes ::= INCLUDES | empty

ValueRange ::= LowerEndpoint ".." UpperEndpoint

LowerEndpoint ::= LowerEndValue | LowerEndValue "<"

UpperEndpoint ::= UpperEndValue | "<" UpperEndValue

LowerEndValue ::= Value | MIN

UpperEndValue ::= Value | MAX

SizeConstraint ::= SIZE Constraint

PermittedAlphabet ::= FROM Constraint

TypeConstraint ::= Type

InnerTypeConstraints ::=
 WITH COMPONENT SingleTypeConstraint |
 WITH COMPONENTS MultipleTypeConstraints

SingleTypeConstraint ::= Constraint

MultipleTypeConstraints ::= FullSpecification | PartialSpecification

FullSpecification ::= "{" TypeConstraints "}"

PartialSpecification ::= "{" "... " "," TypeConstraints "}"

TypeConstraints ::=
 NamedConstraint |
 NamedConstraint "," TypeConstraints

NamedConstraint ::=
 identifiant ComponentConstraint

ComponentConstraint ::= ValueConstraint PresenceConstraint

ValueConstraint ::= Constraint | empty

PresenceConstraint ::= PRESENT | ABSENT | OPTIONAL | empty