UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T X.694

(01/2004)

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

SÉRIE X: RÉSEAUX DE DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

Réseautage OSI et aspects systèmes – Notation de syntaxe abstraite numéro un (ASN.1)

Technologies de l'information – Règles de codage ASN.1: mappage en ASN.1 des définitions de schéma XML du W3C

Recommandation UIT-T X.694

RECOMMANDATIONS UIT-T DE LA SÉRIE X

RÉSEAUX DE DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS

RÉSEAUX PUBLICS DE DONNÉES	
Services et fonctionnalités	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalisation et commutation	X.50–X.89
Aspects réseau	X.90–X.149
Maintenance	X.150–X.179
Dispositions administratives	X.180–X.199
INTERCONNEXION DES SYSTÈMES OUVERTS	
Modèle et notation	X.200–X.209
Définitions 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 des couches	X.280–X.289
Tests de conformité	X.290–X.299
INTERFONCTIONNEMENT DES RÉSEAUX	
Généralités	X.300–X.349
Systèmes de transmission de données par satellite	X.350–X.369
Réseaux à protocole Internet	X.370–X.399
SYSTÈMES DE MESSAGERIE	X.400–X.499
ANNUAIRE	X.500–X.599
RÉSEAUTAGE OSI ET ASPECTS SYSTÈMES	
Réseautage	X.600–X.629
Efficacité	X.630–X.639
Qualité de service	X.640–X.649
Dénomination, adressage et enregistrement	X.650–X.679
Notation de syntaxe abstraite numéro un (ASN.1)	**X.680–X.699**
GESTION OSI	
Cadre général et architecture de la gestion-systèmes	X.700–X.709
Service et protocole de communication de gestion	X.710–X.719
Structure de l'information de gestion	X.720–X.729
Fonctions de gestion et fonctions ODMA	X.730–X.799
SÉCURITÉ	X.800–X.849
APPLICATIONS OSI	
Engagement, concomitance et rétablissement	X.850–X.859
Traitement transactionnel	X.860–X.879
Opérations distantes	X.880–X.899
TRAITEMENT RÉPARTI OUVERT	X.900–X.999
SÉCURITÉ DES TÉLÉCOMMUNICATIONS	X.1000–

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

NORME INTERNATIONALE ISO/CEI 8825-5
RECOMMANDATION UIT-T X.694

Technologies de l'information – Règles de codage ASN.1: mappage en ASN.1 des définitions de schéma XML du W3C

Résumé

La présente Recommandation | Norme internationale définit des règles pour le mappage d'un schéma XSD (un schéma conforme à la spécification du schéma XML du W3C) en un schéma ASN.1 afin d'utiliser les règles de codage de l'ASN.1 telles que les règles de codage de base (BER, *basic encoding rules*), les règles de codage distinctives de la notation ASN.1 (DER, *distinguished encoding rules*), les règles de codage compact de la notation ASN.1 (PER, *packed encoding rules*) ou les règles de codage XML (XER, *XML encoding rules*) pour le transfert des informations défini par le schéma XSD.

L'utilisation de la présente Recommandation | Norme internationale avec les règles de codage XML étendues (EXTENDED-XER) donne la même représentation de valeurs XML que celle définie par le schéma XSD original.

Source

La Recommandation X.694 de l'UIT-T a été approuvée le 13 janvier 2004 par la Commission d'études 17 (2001-2004) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8. Un texte identique est publié comme Norme internationale ISO/CEI 8825-5.

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. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies 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.

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.

Le respect de cette Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité) et considère que la Recommandation est respectée lorsque toutes ces dispositions sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT n'avait pas été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

TABLE DES MATIÈRES

Introduction

La présente Recommandation | Norme internationale spécifie un mappage de définition de schéma XML du W3C (un schéma XSD) vers un schéma ASN.1. Le mappage peut s'appliquer à tout schéma XSD. Il spécifie la création d'un ou plusieurs modules ASN.1 contenant des définitions de type, ainsi que des instructions de règles de codage XML en ASN.1. Elles sont décrites conjointement comme un schéma ASN.1 pour les documents en langage XML.

Ce schéma ASN.1, lorsqu'il est utilisé avec les règles de codage XML étendues en ASN.1 (EXTENDED-XER), peut être utilisé pour générer et valider le même ensemble de documents en langage XML du W3C que le schéma XSD original. Les types et les codages ASN.1 qui en résultent prennent en charge le même contenu sémantique que le schéma XSD d'origine. Et donc, on peut utiliser de façon interchangeable les outils ASN.1 avec les outils XSD pour la création et le traitement des documents spécifiés en langage XML.

On peut utiliser d'autres règles de codage ASN.1 normalisées, telles que les règles de codage distinctives de la notation ASN.1 (DER, *distinguished encoding rules*) ou les règles de codage compact de la notation ASN.1 (PER*, packed encoding rules*) en conjonction avec ce mappage normalisé.

La combinaison de la présente Recommandation | Norme internationale avec les règles de codage ASN.1 donne des codages binaires canoniques et compacts entièrement normalisés et indépendants de leur fabricant pour les données définies qui utilisent un schéma XSD.

Le schéma ASN.1 fait une séparation nette entre la spécification du contenu informatif des messages (leur syntaxe abstraite) et la forme précise du document en langage XML (par exemple, l'utilisation d'attributs au lieu d'éléments). Il en résulte un schéma à la fois plus clair et généralement plus concis que le schéma XSD d'origine.

L'Annexe A fait partie intégrante de la présente Recommandation | Norme internationale et est un module ASN.1 contenant un ensemble d'allocations de types ASN.1 qui correspondent à chacun des types de données XSD incorporés. Le mappage des schémas XSD en schémas ASN.1 importe les noms de référence de type de ces allocations de type ou inclut les définitions de type en ligne.

L'Annexe B ne fait pas partie intégrante de la présente Recommandation | Norme internationale et récapitule les valeurs d'identifiant d'objet allouées dans la présente Recommandation | Norme internationale.

L'Annexe C ne fait pas partie intégrante de la présente Recommandation | Norme internationale et donne des exemples du mappage des schémas XSD en schémas ASN.1.

L'Annexe D ne fait pas partie intégrante de la présente Recommandation | Norme internationale et décrit l'utilisation du mappage défini dans la présente Recommandation | Norme internationale, en conjonction avec les règles de codage ASN.1 normalisées, pour donner des codages canoniques et compacts pour les données définies comme utilisant un schéma XSD.

NORME INTERNATIONALE
RECOMMANDATION UIT-T

Technologies de l'information – Règles de codage ASN.1: mappage en ASN.1 des définitions de schéma XML du W3C

1 Domaine d'application

La présente Recommandation | Norme internationale spécifie un mappage de tout schéma XSD en un schéma ASN.1. Le schéma ASN.1 prend en charge la même sémantique et valide le même ensemble de documents en langage XML.

La présente Recommandation | Norme internationale spécifie les instructions de codage XER finales qui sont à appliquer en tant que partie du mappage défini en types ASN.1, mais ne spécifie pas quelle forme syntaxique doit être utilisée pour la spécification de ces instructions de codage XER finales, ni l'ordre ou la façon dont elles sont allouées.

NOTE – Ceux qui implémentent les outils de création de ces mappages peuvent choisir n'importe quelle forme syntaxique ou ordre d'allocation qui a pour résultat l'application des instructions de codage XER finales spécifiées. Les exemples donnés dans la présente Recommandation | Norme internationale utilisent généralement la forme de type préfixe, mais on peut lui préférer l'utilisation d'une section de contrôle de codage XER pour le mappage d'un schéma XSD complet, ce qui est une question de style.

Il y a différentes façons (syntaxiquement) d'allouer des instructions de codage XER à utiliser dans des codages EXTENDED-XER (par exemple, l'utilisation d'instructions de codage en préfixe de type ASN.1 ou l'utilisation d'une section de contrôle de codage XER). Le choix de ces formes syntaxiques est une question de style et est en dehors du domaine d'application de la présente Recommandation | Norme internationale.

2 Références normatives

Les Recommandations | Normes internationales et spécifications du W3C suivantes contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente Recommandation | Norme internationale. Au moment de la publication, les éditions indiquées étaient en vigueur. Toutes Recommandations, Normes internationales et spécifications du W3C 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 | Normes internationales et spécifications du W3C indiquées ci-après. Les membres de la CEI et de l'ISO possèdent le registre des Normes internationales en vigueur. Le Bureau de la normalisation des télécommunications de l'UIT tient à jour une liste des Recommandations de l'UIT-T en vigueur. Le W3C tient à jour la liste des spécifications du W3C en vigueur. La référence à un document dans la présente Recommandation | Norme internationale ne donne pas à ce document, en tant que tel, le statut d'une Recommandation ou Norme internationale.

2.1 Recommandations | Normes internationales identiques

NOTE – L'ensemble complet des Recommandations | Normes internationales de l'ASN.1 figure ci-dessous, car elles peuvent toutes être applicables pour des utilisations particulières de la présente Recommandation | Norme internationale. Lorsqu'il n'y est pas fait directement référence dans le corps de la présente Recommandation | Norme internationale, un symbole † est ajouté à la référence.

- Recommandation UIT-T X.680 (2002) | ISO/CEI 8824-1:2002, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification de la notation de base.*
- Recommandation UIT-T X.681 (2002) | ISO/CEI 8824-2:2002, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification des objets informationnels*. †
- Recommandation UIT-T X.682 (2002) | ISO/CEI 8824-3:2002, *Technologies de l'information – Notation de syntaxe abstraite numéro un: spécification des contraintes.*
- Recommandation UIT-T X.683 (2002) | ISO/CEI 8824-4:2002, *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 (2002) | ISO/CEI 8825-1:2002, *Technologies de l'information – Règles de codage ASN.1: spécification des règles de codage de base, des règles de codage canoniques et des règles de codage distinctives.* †

- Recommandation UIT-T X.691 (2002) | ISO/CEI 8825-2:2002, *Technologies de l'information – Règles de codage ASN.1): spécification des règles de codage compact.* †
- Recommandation UIT-T X.692 (2002) | ISO/CEI 8825-3:2002, *Technologies de l'information – Règles de codage ASN.1: spécification de la notation de contrôle de codage (ECN).* †
- Recommandation UIT-T X.693 (2001) | ISO/CEI 8825-4:2002, *Technologies de l'information – Règles de codage ASN.1: règles de codage XML (XER).*
- Recommandation UIT-T X.693 (2001)/Am. 1 (2003) | ISO/CEI 8825-4:2002/Amd. 1:2004, *Technologies de l'information – Règles de codage ASN.1: règles de codage XML (XER) – Amendement 1: Prise en charge des règles de codage XML étendues (EXTENDED-XER).*

2.2 Autres références

- ISO 8601:2000, *Data elements and interchange formats – Information interchange – Representation of dates and times.* (Eléments de données et formats d'échange – Echange d'information – Représentation de la date et de l'heure.)
- ISO/CEI 10646-1:2000, *Technologies de l'information – Jeu universel de caractères codés sur plusieurs octets (JUC) – Partie 1*: *Architecture et plan multilingue de base.*
- W3C XML 1.0:2000, *Extensible Markup Language (XML) 1.0 (Second Edition),* (Langage de balisage extensible (XML) *Recommandation du W3C, Copyright © [6 octobre 2000] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut national de recherche en informatique et en automatique, Keio University), http://www.w3.org/TR/2000/REC-xml-20001006.*
- W3C XML Namespaces:1999, *Namespaces in XML,* (Espaces de nommage en langage XML)*, Recommandation du W3C, Copyright © [14 janvier 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut national de recherche en informatique et en automatique, Keio University), http://www.w3.org/TR/1999/REC-xml-names-19990114.*
- W3C XML Information Set:2001, XML *Information Set,* (Ensemble d'informations sur le langage XML) *Recommandation du W3C, Copyright © [24 octobre 2001] World Wide Web Consortium (Massachusetts Institute of Technology, Institut national de recherche en informatique et en automatique, Keio University), http://www.w3.org/TR/2001/REC-xml-infoset-20011024.*
- W3C XML Schema:2001, *XML Schema Part 1: Structures* (Schéma XML, Partie 1: Structures), *Recommandation du W3C, Copyright © [2 mai 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut national de recherche en informatique et en automatique, Keio University), http://www.w3.org/TR/2001/REC-xmlschema-1-20010502*.
- W3C XML Schema:2001, *XML Schema Part 2: Datatypes* (Schéma XML, Partie 2: Types de données)*, Recommandation du W3C, Copyright © [2 mai 2001] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2001/REC-xmlschema-2-20010502*.

  NOTE – Lorsque la référence "Schéma XML du W3C" est utilisée dans la présente Recommandation | Norme internationale, elle se rapporte à la Partie 1 et à la Partie 2 du schéma XML du W3C.
- IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax* (Identificateurs uniformes de ressource (URI): Syntaxe générique).
- IETF RFC 1766 (1995), *Tags for the Identification of Languages* (Etiquettes pour l'identification des langages).

3 Définitions

3.1 Définitions importées

3.1.1 La présente Recommandation | Norme internationale utilise les termes définis dans la Rec. UIT-T X.680 | ISO/CEI 8824-1 et dans la Rec. UIT-T X.693 | ISO/CEI 8825-4.

NOTE – En particulier, les termes "instructions de codage XER finales", "préfixe de type" et "section de contrôle de codage XER" sont définis dans ces Recommandations | Normes internationales.

3.1.2 La présente Recommandation | Norme internationale utilise aussi les termes définis dans le schéma XML du W3C et dans l'ensemble d'informations sur le langage XML du W3C.

NOTE 1 – On estime que ces termes ne sont pas en contradiction avec les termes mentionnés au § 3.1.1. En cas de conflit, c'est la définition du § 3.1.1 qui s'applique.

NOTE 2 – En particulier, les termes "composant de schéma" et "propriété (d'un composant de schéma)" sont définis dans le schéma XML du W3C et le terme "item d'information d'élément" est défini dans l'ensemble d'informations sur le langage XML du W3C.

3.2 Définitions supplémentaires

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

3.2.1 **namespace** (*espace de nommage*) **XSD**: namespace avec un URI de "http://www.w3.org/2001/XMLSchema".

3.2.2 **namespace XSI**: namespace avec un URI de "http://www.w3.org/2001/XMLSchema-instance".

3.2.3 **namespace XML**: namespace avec un URI de "http://www.w3.org/XML/1998/namespace".

4 Abréviations

Pour les besoins de la présente Recommandation | Norme internationale, les abréviations suivantes s'appliquent:

ASN.1 notation de syntaxe abstraite numéro un (*abstract syntax notation one*)
BER règles de codage de base de la notation ASN.1 (*ASN.1 basic encoding rules*)
DER règles de codage distinctives de la notation ASN.1 (*ASN.1 distinguished encoding rules*)
PER règles de codage compact de la notation ASN.1 (*ASN.1 packed encoding rules*)
URI identificateur uniforme de ressource de l'IETF (*IETF uniform resource identifier*)
XER règles de codage XML de la notation ASN.1 (*ASN.1 XML encoding rules*)
XML langage de balisage extensible du W3C (*W3C extensible markup language*)
XSD schéma de langage XML du W3C (*W3C XML schema*)

5 Notation

5.1 La présente Recommandation | Norme internationale se réfère à la notation définie par la Rec. UIT-T X.680 | ISO/CEI 8824-1, la Rec. UIT-T X.682 | ISO/CEI 8824-3, le document XML 1.0 du W3C et le schéma de langage XML du W3C.

5.2 Lorsqu'il est nécessaire de spécifier dans le corps de la présente Recommandation | Norme internationale, formellement ou par des exemples, les allocations d'instructions de codage XER, on utilise généralement la notation par préfixe de type (mais voir les § 6.3 et 6.4). A l'Annexe A est utilisée une section de contrôle de codage XER.

5.3 Dans la présente Recommandation | Norme internationale, la police **`Courier gras`** est utilisée pour la notation ASN.1 et la police **Arial gras** est utilisée pour la notation XSD et pour les termes et concepts XSD.

5.4 Les schémas XSD utilisés dans les exemples de la présente Recommandation | Norme internationale utilisent le préfixe **xsd:** pour identifier l'espace de nommage (*namespace*) XSD.

6 Objet et limites de la normalisation

6.1 Le mappage à l'ASN.1 qui est spécifié dans la présente Recommandation | Norme internationale vérifie que:

a) tout module ASN.1 résultant créé par des outils se conformant à la présente Recommandation | Norme internationale (à partir du même schéma XSD) définit les mêmes valeurs abstraites (structurées);

b) tout codage BASIC-XER, CXER, EXTENDED-XER et binaire de cette spécification ASN.1 résultante produira le même codage (sous réserve des options du codeur); et

c) tout document XML se conformant au schéma XSD source est un codage EXTENDED-XER valide des valeurs abstraites de cette spécification ASN.1.

6.2 De nombreux aspects d'une définition en ASN.1 (tels que l'utilisation d'un espace blanc ou des sections de contrôle de codage ou des préfixes de type) n'affectent ni les valeurs abstraites en cours de définition ni les codages XER ou binaires de ces valeurs. De tels aspects de la définition en ASN.1 ne sont généralement pas normalisés dans la présente Recommandation | Norme internationale.

6.3 En ASN.1, il existe de nombreuses façons différentes d'allouer une instruction de codage XER à un type, y compris:

a) l'utilisation d'un préfixe de type pour chaque instruction de codage à allouer; ou

b) l'utilisation d'une section de contrôle de codage, avec une instruction de codage distincte pour chaque allocation nécessaire; ou

c) l'utilisation d'une section de contrôle de codage, avec une seule instruction de codage faisant une allocation globale, pouvant être complétée par l'utilisation d'une instruction de codage négative pour des types spécifiques.

6.4 La présente Recommandation | Norme internationale spécifie les moments où une instruction de codage XER finale doit être présente et utilise la syntaxe du § 6.3 a) dans la plupart de ses exemples. Cependant, l'utilisation des différentes options données au § 6.3 n'est pas normalisée et les implémentations du mappage conformes (à la présente norme) peuvent choisir toute forme syntaxique (ou une combinaison de formes syntaxiques) pour l'allocation des instructions de codage XER finales.

NOTE – Le choix parmi ces options n'affecte pas les codages binaires ou XML finaux.

6.5 Il n'est pas fourni de spécification formelle du mappage requis.

6.6 La présente Recommandation | Norme internationale ne s'occupe que du mappage des schémas XSD qui se conforment au schéma XML du W3C.

NOTE – Une telle conformité peut résulter de la fourniture d'un ou plusieurs documents selon le schéma XSD du W3C ou par d'autres moyens, comme spécifié dans le schéma XML du W3C.

7 Mappage de schémas XSD

7.1 Un mappage se fonde sur un schéma XSD source, qui est un ensemble de composants de schéma (voir la partie 1, § 2.2 du schéma XML du W3C). Aucune représentation particulière de composants de schéma ou d'ensembles de composants de schéma n'est nécessaire ni supposée pour le mappage, bien qu'on suppose que le schéma XSD source sera habituellement fourni sous forme d'un ou plusieurs documents du schéma XML (voir la partie 1, § 3.15.2 du schéma XML du W3C).

NOTE 1 – Dans la mesure où le mappage est défini en termes de composants de schéma (et non selon leur représentation XML), il n'est pas affecté par les détails de la représentation XML, comme l'utilisation de documents à schéma multiple liés par les items d'information d'élément **xsd:include** et **xsd:redefine**, le placement d'items d'information d'élément dans l'un ou l'autre des documents de schéma, l'ordre des items d'information d'élément **xsd:attribute** dans un item d'information d'élément **xsd:complexType** et ainsi de suite.

NOTE 2 – Deux ensembles de documents de schéma qui diffèrent par de nombreux aspects mais représentent le même ensemble de composants de schéma génèrent le même ensemble d'allocations de type en ASN.1, avec les mêmes instructions de codage finales allouées à eux et à leurs composants à toute profondeur.

7.2 Le schéma XSD source doit satisfaire à toutes les contraintes imposées par la spécification XSD. Si le schéma XSD source est représenté (en partie ou en totalité) comme un ensemble de documents de schéma XML, chaque document de schéma doit être valide par rapport au schéma XSD pour les schémas (voir la partie 1, Appendice A du schéma XML du W3C).

7.3 Au moins un module ASN.1 (voir le § 7.4) doit être généré pour chaque **namespace cible** (**target namespace**) différent (que ce soit un nom de namespace ou le mot clé **absent**) qui est le **namespace cible** d'un ou plusieurs composants de schéma dans le schéma XSD source. Chaque module ASN.1 doit contenir une ou plusieurs allocations de type correspondant aux composants de schéma de niveau supérieur (voir le § 7.9) qui ont le même **namespace cible**. Chaque module ASN.1 peut aussi contenir une ou plusieurs allocations spéciales de type ASN.1 dont les allocations de type ASN.1 associées sont dans le même module ASN.1 (voir le § 7.6).

NOTE – Les composants de schéma représentés dans les documents de schéma multiple sont incorporés au même schéma XSD grâce à l'utilisation des items d'information d'élément **xsd:include, xsd:redefine** et **xsd:import**.

7.4 Le nombre de modules ASN.1 générés pour chaque **namespace cible** (y compris le mot clé **absent**) peut être supérieur à un, mais aucun module ASN.1 ne doit contenir d'allocation de type correspondant aux composants de schéma de niveau supérieur avec des **namespace**s **cible** différents (y compris le mot clé **absent**).

7.5 Lorsque plusieurs modules ASN.1 sont générés pour un **namespace cible** donné (y compris le mot clé **absent**), toutes les allocations de type qui y sont présentes doivent être générées comme si elles étaient ajoutées à un module ASN.1 unique dans le but de générer des noms de référence de type distincts (voir le § 10.3). Les noms de référence de type générés à partir des **name**s (*noms*) des composants de schéma du niveau supérieur avec un **namespace cible** donné doivent avoir les mêmes noms de référence de type sans considération du nombre de modules ASN.1 générés pour ce **namespace cible** et sans considération de la façon dont les allocations de type sont réparties parmi les divers modules ASN.1.

NOTE – Ceci est destiné à donner de la souplesse sans compromettre l'interopérabilité.

7.6 Chaque allocation de type ASN.1 spéciale (voir les § 29, 30 et 31) doit être insérée dans le même module ASN.1 que celui de son allocation de type ASN.1 associée (voir respectivement les § 29.4, 31.4 et 30.4).

7.7 Tous les modules ASN.1 générés par mappage doivent contenir (dans la section de contrôle de codage XER) une instruction de codage **`GLOBAL-DEFAULTS MODIFIED-ENCODINGS`** et une instruction de codage **`GLOBAL-DEFAULTS CONTROL-NAMESPACE`** spécifiant le namespace XSI.

7.8 Un schéma XSD source doit être traité comme suit:

a) pour chaque **element declaration** (*déclaration d'élément*) de niveau supérieur, une allocation de type ASN.1 doit être générée en appliquant le § 14 à l'**element declaration**;

b) pour chaque **attribute declaration** (*déclaration d'attribut*) de niveau supérieur, une allocation de type ASN.1 doit être générée en appliquant le § 15 à l'**attribute declaration**;

c) pour chaque **simple type definition** (*définition de type simple*) de niveau supérieur définie par l'usager, une allocation de type ASN.1 doit être générée en appliquant le § 13 à la **simple type definition**;

d) pour chaque **complex type definition** (*définition de type complexe*) de niveau supérieur, une allocation de type ASN.1 doit être générée en appliquant le § 20 à la **complex type definition**;

e) pour chaque **model group definition** (*définition de groupe modèle*) dont **model group** (*groupe modèle*) a un **compositor** (*composeur*) de **sequence** (*séquence*) ou **choice** (*choix*), une allocation de type ASN.1 doit être générée en appliquant le § 17 à la **model group definition**.

NOTE 1 – Les composants de schéma restants du schéma XSD de source seront traités comme résultat du mappage de ces composants de schéma.

NOTE 2 – L'ordre dans lequel les composants de schéma doivent être mappés est spécifié au § 10.4. L'ordre des items de la liste ci-dessus n'a pas de signification pour le mappage.

7.9 La colonne 1 du Tableau 1 donne la liste des composants de schéma. La colonne 2 donne la référence au paragraphe du schéma XML du W3C qui définit le composant de schéma. La colonne 3 donne la liste des paragraphes qui définissent le mappage de ces composants de schéma en ASN.1.

Tableau 1 – Mappage des composants de schéma XSD

Composant de schéma XSD	Référence du schéma XML du W3C	Mappage défini
attribute declaration	Partie 1, 3.2	§ 15
element declaration	Partie 1, 3.3	§ 14
complex type definition	Partie 1, 3.4	§ 20
attribute use	Partie 1, 3.5	§ 22
attribute group definition	Partie 1, 3.6	*non mappé comme tel*
model group definition	Partie 1, 3.7	§ 17
model group	Partie 1, 3.8	§ 18
particle	Partie 1, 3.9	§ 19
wildcard	Partie 1, 3.10	§ 21
identity-constraint definition	Partie 1, 3.11	*ignoré par le mappage*
notation declaration	Partie 1, 3.12	*ignoré par le mappage*
annotation	Partie 1, 3.13	*ignoré par le mappage*
simple type definition	Partie 1, 3.14	§ 11, 13
schema	Partie 1, 3.15	§ 9
ordered	Partie 2, 4.2.2.1	*ignoré par le mappage*
bounded	Partie 2, 4.2.3.1	*ignoré par le mappage*
cardinality	Partie 2, 4.2.4.1	*ignoré par le mappage*
numeric	Partie 2, 4.2.5.1	*ignoré par le mappage*
length	Partie 2, 4.3.1.1	§ 12
minLength	Partie 2, 4.3.2.1	§ 12
maxLength	Partie 2, 4.3.3.1	§ 12
pattern	Partie 2, 4.3.4.1	§ 12

Tableau 1 – Mappage des composants de schéma XSD

Composant de schéma XSD	Référence du schéma XML du W3C	Mappage défini
enumeration	Partie 2, 4.3.5.1	§ 12
whiteSpace	Partie 2, 4.3.6.1	§ 12
maxInclusive	Partie 2, 4.3.7.1	§ 12
maxExclusive	Partie 2, 4.3.8.1	§ 12
minExclusive	Partie 2, 4.3.9.1	§ 12
minInclusive	Partie 2, 4.3.10.1	§ 12
totalDigits	Partie 2, 4.3.11.1	§ 12
fractionDigits	Partie 2, 4.3.12.1	§ 12

8 Composants de schéma et propriétés ignorés

8.1 Le mappage doit ignorer les composants de schéma et les propriétés dont la liste figure dans le présent paragraphe.

8.2 Toutes les **annotation**s (voir la Partie 1, § 3.13 du schéma XML du W3C) doivent être ignorées.

NOTE – Tous les items d'information d'attribut dans un document de schéma avec des noms qualifiés avec des namespaces autres que le namespace XSD (voir la Partie 1, § 3.13.1 du schéma XML du W3C) ont une propriété d'**annotation**s et sont ignorés.

8.3 Toutes les **identity-constraint definition**s (voir la Partie 1, § 3.11 du schéma XML du W3C) doivent être ignorées.

NOTE – L'**identity-constraint definition** (*définition de contrainte d'identité*) fournit des mécanismes pour spécifier les contraintes de référentiel qui peuvent être nécessaires dans une instance valide. L'ASN.1 n'a pas actuellement de concept pour de telles contraintes et celles-ci ne peuvent donc être mappées dans une spécification ASN.1 formelle, mais elles peuvent être incluses comme commentaires normatifs qui ont un caractère obligatoire pour une implémentation d'application.

8.4 Toutes les **notation declaration**s (*déclaration de notation*) (voir la Partie 1, § 3.12 du schéma XML du W3C) doivent être ignorées.

8.5 Tous les composants de schéma qui sont les **fundamental facet**s (*facettes fondamentales*) (**ordered** (*ordonné*), **bounded** (*attaché*), **cardinality** (*cardinalité*), **numeric** (*numérique*)) des **définition**s **de type simple** (voir la Partie 2, § 4.2 du schéma XML du W3C) doivent être ignorés.

8.6 Les propriétés **identity-constraint definition**s, **substitution group exclusion**s (*exclusion de groupe de substitution*) et **disallowed substitution**s (*substitution non autorisée*) des **déclaration**s **d'élément** doivent être ignorées.

8.7 Les propriétés **final** (*finale*), **abstract** (*abstraite*) et **prohibited substitution**s (*substitution interdite*) des **définition**s **de type complexe** doivent être ignorées.

8.8 La propriété **process contents** (*traiter les contenus*) des **wildcard**s (*caractères substitution*) doit être ignorée.

NOTE – Il n'y a pas de prise en compte en ASN.1 d'action autre que **skip**.

8.9 Les propriétés **fundamental facet**s et **final** des **définition**s **de type simple** doivent être ignorées.

8.10 Toutes les **value constraint**s (*contraintes de valeur*) qui sont présentes sur toutes les **déclaration**s **d'élément** ou **déclaration**s **d'attribut** dont la **définition de type** est **xsd:QName** ou une **définition de type simple** déduite de **xsd:QName** ou **xsd:NOTATION** doivent être ignorées.

8.11 Toutes les **attribute group definition**s *(définition de groupe d'attributs)* doivent être ignorées.

NOTE – Les **attribute use**s *(utilisations d'attribut)* dans une **attribute group definition** sont incorporées dans les **attribute use**s des **complex type definition**s dont la représentation XML contient une référence à l'**attribute group definition**.

9 Les modules et namespaces ASN.1

NOTE – Une description complète des relations entre le concept de namespace XML du W3C et le nommage en ASN.1 est donnée dans la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 16. Les noms et les identificateurs de référence de type définis dans un module ASN.1 se voient attribuer un namespace au moyen d'une instruction de codage **NAMESPACE**, faute de quoi ils n'ont pas de namespace. Le mappage génère des instructions de codage **NAMESPACE** en tant que de besoin.

9.1 Le mappage génère un ou plusieurs modules ASN.1 correspondant à tous les composants de schéma qui ont le même **namespace cible** (*target*, cible) dans le schéma.

9.2 Le "ModuleIdentifier" ASN.1 (voir la Rec. UIT-T X.680 | ISO/CEI 8824-1, § 12) que doit générer le mappage n'est pas normalisé. Lorsque des déclarations **IMPORTS** sont utilisées, les noms de module et les identifiants de module ASN.1 dans les déclarations **IMPORTS** doivent être ceux générés pour les modules ASN.1 générés par le mappage.

NOTE – Le choix du "ModuleIdentifier" n'affecte pas le codage dans les règles de codage standard.

9.3 Les modules ASN.1 doivent avoir un "TagDefault" de **AUTOMATIC TAGS**.

9.4 Dans chaque module ASN.1, il doit y avoir une déclaration **IMPORTS** ASN.1 qui importe les noms de référence de type ASN.1 dans le module nommé **XSD** spécifié dans l'Annexe A qui sont référencés dans le module ASN.1.

9.5 Les déclarations **IMPORTS** doivent aussi importer les noms de référence de type ASN.1 d'allocation de type qui ont été placés (par suite du mappage) dans d'autres modules ASN.1 mais sont référencés dans ce module ASN.1.

9.6 Il ne doit pas y avoir de déclaration **EXPORTS**.

NOTE – Ceci signifie que tous les noms de référence de type ASN.1 dans le module ASN.1 peuvent être importés dans d'autres modules.

10 Conversion de nom

10.1 Généralités

10.1.1 La présente Recommandation | Norme internationale spécifie la génération de:

a) noms de référence de type ASN.1 correspondant aux **name**s de **définitions de groupe modèle**, **déclaration**s **d'élément** de niveau supérieur, **déclaration**s **d'attribut** de niveau supérieur, **définition**s **de type complexe** de niveau supérieur et **définition**s **de type simple** de niveau supérieur définies par l'usager;

b) identifiants ASN.1 correspondant aux **name**s de **déclaration**s **d'élément** de niveau supérieur, **déclaration**s **d'attribut** de niveau supérieur, **déclaration**s **d'élément** locales et **déclaration**s **d'attribut** locales;

c) identifiants ASN.1 pour le mappage de certaines **définition**s **de type simple** avec une facette **énumération** (voir les § 12.4.1 et 12.4.2);

d) noms de référence de type ASN.1 d'allocations de type spéciales (voir les § 29, 30 et 31); et

e) identifiants ASN.1 de certains composants de séquence introduits par le mappage (voir § 20).

10.1.2 Tous ces noms ASN.1 sont générés en appliquant le § 10.3 soit au **nom** du composant de schéma correspondant, soit à un membre de la **valeur** d'une facette **énumération** ou d'une chaîne de caractères spécifiée, comme spécifié dans les paragraphes pertinents de la présente Recommandation | Norme internationale.

10.2 Génération des définitions de type ASN.1 qui sont des références aux allocations de type ASN.1

10.2.1 Ce paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une définition de type ASN.1 qui soit une référence (un "DefinedType") à une allocation de type ASN.1.

10.2.2 Si une définition de type ASN.1 (disons R) qui est un "DefinedType" doit être insérée dans un module ASN.1 (disons M) autre que le module ASN.1 où l'allocation de type ASN.1 référencée (disons TA) doit être insérée et que le nom de référence de type de TA est identique au nom de référence de type d'une autre allocation de type ASN.1 en cours d'insertion dans le module M ou à un autre nom de référence de type en cours d'importation dans le module M, R doit alors être une "ExternalTypeReference" (construite comme il convient pour le module M) pour TA, autrement, elle doit être une "typereference" pour TA.

10.3 Génération des identifiants et noms de référence de type

10.3.1 Ce paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer un nom de référence de type ou identifiant ASN.1.

10.3.2 Les **nom**s des **déclaration**s **d'attribut**, des **déclaration**s **d'élément**, des **définition**s **de modèle de groupe**, des **définition**s **de type simple** de niveau supérieur définies par l'usager et les **définition**s **de type complexe** de niveau supérieur peuvent être identiques aux mots réservés ASN.1 ou peuvent contenir des caractères non admis dans les identifiants ASN.1 ou

dans les noms de référence de type ASN.1. De plus, il y a des cas dans lesquels les noms ASN.1 doivent nécessairement être distincts lorsqu'il est permis que les **nom**s des composants de schéma XSD correspondants (d'où sont mappés les noms ASN.1) soient identiques.

10.3.3 Les transformations suivantes doivent être appliquées, dans l'ordre, pour chaque chaîne de caractères mappée en nom ASN.1, lorsque chaque transformation (excepté la première) est appliquée au résultat de la transformation précédente:

- les caractères " " (ESPACE), "**.**" (POINT) et "**_**" (SOULIGNE) doivent tous être remplacés par un "-" (TRAIT D'UNION); et
- tous les caractères sauf "A" à "Z" (LETTRE LATINE A MAJUSCULE à LETTRE LATINE Z MAJUSCULE), "a" à "z" (LETTRE LATINE A MINUSCULE à LETTRE LATINE Z MINUSCULE), "0" à "9" (CHIFFRE ZERO à CHIFFRE NEUF) et "-" (TRAIT D'UNION) doivent être retirés; et
- une séquence de deux caractères ou plus TRAIT D'UNION doit être remplacée par un TRAIT D'UNION unique; et
- des caractères TRAIT D'UNION survenant au début ou à la fin du nom doivent être retirés; et
- si une chaîne de caractères qui doit être utilisée comme nom de référence de type commence par une minuscule, la première lettre doit être mise en majuscule (convertie en majuscule); si elle commence par un chiffre (CHIFFRE ZERO à CHIFFRE NEUF), elle doit être munie en préfixe du caractère "**X**" (LETTRE LATINE X MAJUSCULE); et
- si une chaîne de caractères qui doit être utilisée comme identifiant commence par une lettre majuscule, la première lettre doit être convertie en minuscule; si elle commence par un chiffre (CHIFFRE ZERO à CHIFFRE NEUF), elle doit être munie en préfixe du caractère "**x**" (LETTRE LATINE X MINUSCULE); et
- si une chaîne de caractères qui doit être utilisée comme nom de référence de type est vide, elle doit être remplacée par "**X**" (LETTRE LATINE X MAJUSCULE); et
- si une chaîne de caractères qui doit être utilisée comme identifiant est vide, elle doit être remplacée par "**x**" (LETTRE LATINE X MINUSCULE).

10.3.4 Selon la sorte de nom qui est généré, on applique un des trois paragraphes suivants.

10.3.4.1 Si le nom qu'on génère est le nom de référence de type d'une allocation de type ASN.1 et si la chaîne de caractères générée selon le § 10.3.3 est identique au nom de référence de type d'une autre allocation de type ASN.1 précédemment générée dans le même module ASN.1 ou dans un autre module ASN.1 avec le même namespace (y compris l'absence de namespace) ou est un des mots réservés spécifiés dans la Rec. UIT-T X.680 | ISO/CEI 8824-1, au § 11.27, un suffixe doit alors être ajouté à la chaîne de caractères générée selon le § 10.3.3. Le suffixe doit consister en un TRAIT D'UNION suivi par la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) d'un entier. Cet entier doit être le plus petit entier positif tel que le nouveau nom soit différent du nom de référence de type de toute autre allocation de type ASN.1 précédemment générée dans un de ces modules ASN.1.

10.3.4.2 Si le nom qu'on génère est l'identifiant d'un composant de type de séquence, d'ensemble ou de choix et que la chaîne de caractères générée selon le § 10.3.3 est identique à l'identifiant d'un composant précédemment généré du même type de séquence, ensemble ou choix, un suffixe doit alors être ajouté à la chaîne de caractères générée selon le § 10.3.3. Le suffixe doit consister en un TRAIT D'UNION suivi par la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) d'un entier. Cet entier doit être le plus petit entier positif tel que le nouveau nom soit différent du nom de référence de type de toute autre allocation de type ASN.1 précédemment générée de ce type de séquence, ensemble ou choix.

10.3.4.3 Si le nom qu'on génère est l'"identifiant" dans une "EnumerationItem" d'un type énuméré et si la chaîne de caractères générée selon le § 10.3.3 est identique à l'"identifiant" dans un autre "EnumerationItem" précédemment généré dans le même type énuméré, un suffixe doit alors être ajouté à la chaîne de caractères générée selon le § 10.3.3. Le suffixe doit consister en un TRAIT D'UNION suivi par la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) d'un entier. Cet entier doit être le plus petit entier positif tel que le nouvel identifiant soit différent de l'"identifiant" dans tout autre "EnumerationItem" déjà présent dans ce type énuméré ASN.1.

10.3.5 Pour un nom de référence de type (ou identifiant) ASN.1 qui est généré en appliquant ce § 10.3 au **nom** d'une **déclaration d'élément, déclaration d'attribut, définition de type complexe** de niveau supérieur ou **définition de type simple** de niveau supérieur définie par l'usager, si le nom de référence de type (ou identifiant) généré est différent du **nom**, une instruction de codage **NAME** finale doit être attribuée à l'allocation de type ASN.1 avec ce nom de référence de type (ou au composant qui a cet identifiant) comme spécifié dans un des trois paragraphes qui suivent.

10.3.5.1 Si la seule différence est la taille de la première lettre (qui est une majuscule dans le nom de référence de type et une minuscule dans le **nom**), alors le "Mot-clé" dans l'instruction de codage **NAME** doit être en **MINUSCULE**.

10.3.5.2 Si la seule différence est la taille de la première lettre (qui est une minuscule dans l'identifiant et une majuscule dans le **nom**), alors le "Keyword" (*Mot-clé*) dans l'instruction de codage **NAME** doit être en **MAJUSCULE**.

10.3.5.3 Autrement, le "NewName" (*NouveauNom*) dans l'instruction de codage **NAME** doit être le **nom**.

EXEMPLE – La **définition de type complexe** de niveau supérieur:

```
<xsd:complexType name="COMPONENTS">
        <xsd:sequence>
                <xsd:element name="Elem" type="xsd:boolean"/>
                <xsd:element name="elem" type="xsd:integer"/>
                <xsd:element name="Elem-1" type="xsd:boolean"/>
                <xsd:element name="elem-1" type="xsd:integer"/>
        </xsd:sequence>
</xsd:complexType>
```

est mappée en cette allocation de type ASN.1:

```
COMPONENTS-1 ::= [NAME AS "COMPONENTS"] SEQUENCE {
      elem     [NAME AS CAPITALIZED] BOOLEAN,
      elem-1   [NAME AS "elem"] INTEGER,
      elem-1-1 [NAME AS "Elem-1"] BOOLEAN,
      elem-1-2 [NAME AS "elem-1"] INTEGER }
```

10.3.6 Pour un nom de référence de type (ou identifiant) ASN.1 qui est généré en appliquant ce § 10.3 au **nom** d'une **déclaration d'élément, déclaration d'attribut, définition de type complexe** de niveau supérieur ou **définition de type simple** de niveau supérieur définie par l'usager, si le **namespace cible** du composant de schéma n'est pas **absent**, une instruction de codage **NAMESPACE** finale doit être attribuée à l'allocation de type ASN.1 avec ce nom de référence de type (ou au type nommé avec cet identifiant) et doit spécifier le **namespace cible** du composant de schéma.

10.3.7 Pour un identifiant ASN.1 qui est généré selon ce § 10.3 pour le mappage d'une **définition de type simple** avec une facette **enumeration** où l'identifiant généré est différent du membre correspondant de la **valeur** de la facette d'**enumeration**, une instruction de codage **TEXT** finale doit être attribuée au type énuméré ASN.1, avec des informations qualificatives spécifiant l'"identifiant" dans l'"EnumerationItem" du type énuméré. On applique un des deux paragraphes suivants.

10.3.7.1 Si la seule différence est la taille de la première lettre (qui est en minuscule dans l'identifiant et en majuscule dans le membre de la **valeur** de la facette **enumeration**), le "Mot-clé" dans l'instruction de codage **TEXT** doit être en **MAJUSCULE.**

10.3.7.2 Autrement, le "NewName" dans l'instruction de codage **TEXT** doit être le membre de la **valeur** de la facette **enumeration**.

10.4 Ordre du mappage

10.4.1 Un ordre est imposé aux composants de schéma de niveau supérieur du schéma XSD de source sur lequel le mappage est effectué. Cela s'applique aux **définition**s **de groupe modèle**, aux **définition**s **de type complexe** de niveau supérieur, aux **définition**s **de type simple** de niveau supérieur définies par l'usager, aux **déclaration**s **d'attribut** de niveau supérieur et aux **déclaration**s **d'élément** de niveau supérieur.

NOTE – D'autres composants de schéma de niveau supérieur ne sont pas mappés en ASN.1 et les types de données préconstruits en XSD sont mappés d'une façon spéciale.

10.4.2 L'ordre est spécifié dans les trois paragraphes suivants.

10.4.2.1 Les composants de schéma de niveau supérieur doivent d'abord être ordonnés selon leur **namespace cible,** avec le namespace **absent** qui précède tous les noms de namespace dans l'ordre lexicographique ascendant.

10.4.2.2 Dans chaque namespace cible, les composants de schéma doivent être divisés en quatre ensembles ordonnés comme suit:

a) **déclaration**s **d'élément;**

b) **déclaration**s **d'attribut**;

c) **définition**s **de type complexe** et **définition**s **de type simple**;

d) **définition**s **de groupe modèle**.

10.4.2.3 Dans chaque ensemble (voir le § 10.4.2.2), les composants de schéma doivent être ordonnés par **nom** dans l'ordre lexicographique ascendant.

10.4.3 Le mappage génère quelques allocations de type ASN.1 qui ne correspondent pas directement à un composant de schéma XSD. Ce sont:

a) les types de choix (avec une instruction de codage **`USE-TYPE`** finale) correspondant à une hiérarchie de dérivation de type; les noms de référence de type de ces types ont un suffixe "**`-derivations`**" (voir le § 29);

b) les types de choix (avec une instruction de codage **`USE-TYPE`** finale sur le type et une instruction de codage **`USE-NIL`** finale sur chaque terme de l'alternative) correspondant à une hiérarchie de dérivation de type où la **définition de type simple** de niveau supérieur définie par l'usager ou **la définition de type complexe** qui est la racine de la hiérarchie de dérivation est utilisée comme la **définition de type** d'une ou plusieurs **déclaration**s **d'élément** qui sont **nillable**s (réductibles à néant); les noms de référence de type de ces types ont un suffixe "**`-deriv-nillable`**" (voir le § 29);

c) les types de choix (avec une instruction de codage **`USE-TYPE`** finale sur le type et des instructions de codage **`DEFAULT-FOR-EMPTY`** finales sur chaque terme de l'alternative) correspondant à une hiérarchie de dérivation de type où la **définition de type simple** de niveau supérieur définie par l'usager ou **la définition de type complexe** qui est la racine de la hiérarchie de dérivation est utilisée comme la **définition de type** d'une ou plusieurs **déclaration**s **d'élément** qui ne sont pas **nillable**s et ont une **contrainte de valeur** qui est une valeur par **défaut**; les noms de référence de type de ces types ont un suffixe "**`-deriv-default-`**" (voir le § 29);

d) les types de choix (avec une instruction de codage **`USE-TYPE`** finale sur le type et des instructions de codage **`DEFAULT-FOR-EMPTY`** finales sur chaque terme de l'alternative) correspondant à une hiérarchie de dérivation de type où la **définition de type simple** de niveau supérieur définie par l'usager ou **la définition de type complexe** qui est la racine de la hiérarchie de dérivation est utilisée comme la **définition de type** d'une ou plusieurs **déclaration**s **d'élément** qui ne sont pas **nillable**s et ont une **contrainte de valeur** qui est une valeur **fixe;** les noms de référence de type de ces types ont un suffixe "**`-deriv-fixed-`**" (voir le § 29);

e) les types de choix (avec une instruction de codage **`USE-TYPE`** finale sur le type et des instructions de codage **`USE-NIL`** et **`DEFAULT-FOR-EMPTY`** finales sur chaque terme de l'alternative) correspondant à une hiérarchie de dérivation de type où la **définition de type simple** de niveau supérieur définie par l'usager ou la **définition de type complexe** qui est la racine de la hiérarchie de dérivation est utilisée comme la **définition de type** d'une ou plusieurs **déclaration**s **d'élément** qui sont **nillable**s et ont une **contrainte de valeur** qui est une valeur par **défaut;** les noms de référence de type de ces types ont un suffixe "**`-deriv-nillable-default-`**" (voir le § 29);

f) les types de choix (avec une instruction de codage **`USE-TYPE`** finale sur le type et des instructions de codage **`USE-NIL`** et **`DEFAULT-FOR-EMPTY`** finales sur chaque terme de l'alternative) correspondant à une hiérarchie de dérivation de type où la **définition de type simple** de niveau supérieur définie par l'usager ou la **définition de type complexe** qui est la racine de la hiérarchie de dérivation est utilisée comme la **définition de type** d'une ou plusieurs **déclaration**s **d'élément** qui sont **nillable**s et ont une **contrainte de valeur** qui est une valeur **fixe**; les noms de référence de type de ces types ont un suffixe "**`-deriv-nillable-fixed-`**" (voir le § 29);

g) les types de choix (avec une instruction de codage **`UNTAGGED`** finale) correspondant à un groupe de substitution d'élément; les noms de référence de type de ces types ont un suffixe "**`-group`**" (voir le § 31);

h) les types de séquence (avec une instruction de codage **`USE-NIL`** finale) correspondant à l'utilisation d'une **définition de type simple** de niveau supérieur définie par l'usager ou la **définition de type complexe** comme la **définition de type** d'une ou plusieurs **déclaration**s **d'élément** qui sont **nillable**s; les noms de référence de type de ces types ont un suffixe "**`-nillable`**" (voir le § 30).

10.4.4 Toutes les allocations de type ASN.1 qui correspondent directement aux composants de schéma XSD dans le schéma XSD source doivent être générées avant toute allocation de type ASN.1 dont la liste figure au § 10.4.3 (s'il en est).

10.4.5 Les allocations de type ASN.1 qui correspondent directement aux composants de schéma XSD doivent être générées dans l'ordre des composants de schéma XSD correspondants (voir le § 10.4.1). Les allocations de type ASN.1 dont la liste figure au § 10.4.3 (s'il en est) doivent être générées dans l'ordre des composants de schéma XSD (voir le § 10.4.1) correspondant à l'"allocation de type associé" (voir § 29, 30 et 31).

10.4.6 Pour les alinéas c) à f) du § 10.4.3, si la **définition de type simple** ou la **définition de type complexe** qui est la racine de la hiérarchie de dérivation est utilisée comme la **définition de type** de **déclaration**s **d'élément**s multiples qui ont des valeurs différentes dans la **contrainte de valeur**, les allocations de type ASN.1 doivent être générées dans l'ordre lexicographique ascendant de la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) de la valeur dans la **contrainte de valeur.**

11 Utilisations des mappages des datatypes incorporés en XSD

11.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une définition de type ASN.1 correspondant à l'utilisation d'un datatype incorporé en XSD.

11.2 Une utilisation de datatype incorporé en XSD doit être mappée en une définition de type ASN.1 conformément au Tableau 2. Le tableau donne la définition de type ASN.1 à utiliser. La notation "XSD.Name" indique que la définition de type ASN.1 doit être la définition de type ASN.1 (un "DefinedType") générée en appliquant le § 10.2 à l'allocation de type ASN.1 correspondante présente dans le module **XSD**.

Tableau 2 – Définitions de type ASN.1 correspondant aux utilisations des datatypes incorporés en XSD

Datatype incorporé en XSD	Définition de type ASN.1	Datatype incorporé en XSD	Définition de type ASN.1
anyURI	XSD.AnyURI	int	XSD.Int
anySimpleType	XSD.AnySimpleType	integer	INTEGER
anyType	XSD.AnyType	language	XSD.Language
base64Binary	[BASE64] OCTET STRING	long	XSD.Long
boolean	BOOLEAN	Name	XSD.Name
byte	INTEGER (-128..127)	NCName	XSD.NCName
date	XSD.Date	negativeInteger	INTEGER (MIN..-1)
dateTime	XSD.DateTime	NMTOKEN	XSD.NMTOKEN
decimal	XSD.Decimal	NMTOKENS	XSD.NMTOKENS
double	XSD.Double	nonNegativeInteger	INTEGER (0..MAX)
duration	XSD.Duration	nonPositiveInteger	INTEGER (MIN..0)
ENTITIES	XSD.ENTITIES	normalizedString	XSD.NormalizedString
ENTITY	XSD.ENTITY	NOTATION	XSD.NOTATION
float	XSD.Float	positiveInteger	INTEGER (1..MAX)
gDay	XSD.GDay	QName	XSD.QName
gMonth	XSD.GMonth	short	XSD.Short
gMonthDay	XSD.GMonthDay	string	XSD.String
gYear	XSD.GYear	time	XSD.Time
gYearMonth	XSD.GYearMonth	token	XSD.Token
hexBinary	OCTET STRING	unsignedByte	INTEGER (0..255)
ID	XSD.ID	unsignedInt	XSD.UnsignedInt
IDREF	XSD.IDREF	unsignedLong	XSD.UnsignedLong
IDREFS	XSD.IDREFS	unsignedShort	XSD.UnsignedShort

12 Mappage de facettes

Ce paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour mapper une **facette** d'une **définition de type simple**. Une **facette** d'une **définition de type simple** STD est mappée en une contrainte ASN.1 appliquée à la définition de type ASN.1 correspondant à STD, à moins que STD n'ait une facette **enumeration** qui soit mappée en une "Enumeration" ASN.1 (voir les § 12.4.1 et 12.4.2). Dans ce cas, il n'est pas généré de contrainte ASN.1 à partir de la facette (voir les § 12.1.2, 12.2.1, 12.3.1 et 12.5.1).

12.1 Les facettes length, minLength et maxLength

12.1.1 Les facettes **length** (longueur), **minLength** (longueur minimum) et **maxLength** (longueur maximum) doivent être ignorées pour les datatypes incorporés en XSD **xsd:QName** et **xsd:NOTATION** et pour toute **définition de type simple** qui en est déduite par restriction.

12.1.2 Si une facette **length**, **minLength** ou **maxLength** appartient à une **définition de type simple** qui a aussi une facette **enumeration** en cours de mappage en une "Enumeration" ASN.1 (voir les § 12.4.1 et 12.4.2), aucun "EnumerationItem"

ne doit être inclus dans l'"Enumeration" pour les membres (s'il en est) de la **valeur** de la facette **enumeration** qui ne satisfait pas la facette **length**, **minLength** ou **maxLength**.

12.1.3 Autrement, les facettes **length**, **minLength et maxLength** de la **définition de type simple** doivent être mappées en contrainte de taille ASN.1 conformément au Tableau 3.

Tableau 3 – Contraintes de taille ASN.1 correspondant aux facettes length, minLength et maxLength

Facette XSD	Contrainte de taille ASN.1
length=*valeur*	**(SIZE(***valeur***))**
minLength=*min*	**(SIZE(***min* **.. MAX))**
maxLength=*max*	**(SIZE(0 ..** *max***))**
minLength=*min* **maxLength=***max*	**(SIZE(***min* **..** *max***))**

12.2 La facette pattern

12.2.1 Si une facette **pattern** appartient à une **définition de type simple** qui a aussi une facette **enumeration** en cours de mappage en une "Enumeration" ASN.1 (voir les § 12.4.1 et 12.4.2), aucun "EnumerationItem" ne doit alors être inclus dans l'"Enumeration" pour les membres (s'il en est) de la **valeur** de la facette **enumeration** qui ne satisfait pas à la facette **pattern**.

12.2.2 Autrement, la facette **pattern** doit être mappée en une contrainte définie par l'usager. Un des deux paragraphes suivants s'applique.

12.2.2.1 Si la **valeur** de la facette **pattern** est une expression régulière unique, la contrainte définie par l'usager doit être:

```
(CONSTRAINED BY {/* représentation XML du pattern XSD "xyz" */})
```

où "xyz" est la représentation XML de la **valeur** de la facette **pattern**, sauf que si la sous-chaîne "***/**" apparaît dans la **valeur** de la facette **pattern**, elle doit être remplacée par la chaîne de caractères "***/**".

12.2.2.2 Si la **valeur** de la facette **pattern** est une conjonction d'unions d'expressions régulières (le cas général), la contrainte définie par l'usager n'est pas spécifiée (mais voir le § 12.5.4).

12.3 La facette whiteSpace

12.3.1 Si une facette **whiteSpace** (espace blanc) avec une **valeur** de **replace** ou **collapse** appartient à une **définition de type simple** qui a aussi une facette **enumeration** en cours de mappage en une "Enumeration" ASN.1 (voir les § 12.4.1 et 12.4.2), aucun "EnumerationItem" ne doit être inclus dans l'"Enumeration" pour les membres (s'il en est) de la **valeur** de la facette **enumeration** qui contiennent l'un des caractères TABULATION HORIZONTALE, NOUVELLE LIGNE ou RETOUR CHARIOT ou (dans le cas de **collapse**) contiennent des caractères ESPACE consécutifs, avant, après ou multiples.

12.3.2 Autrement, au plus un des trois paragraphes suivants s'applique:

12.3.2.1 Si la **valeur** de la facette **whiteSpace** est **preserve**, la facette **whiteSpace** doit alors être ignorée.

12.3.2.2 Si la **valeur** de la facette **whiteSpace** est **replace** (remplacer) et si la définition de type ASN.1 correspondant à la **définition de type simple** est un type de chaîne de caractère restreint à l'ASN.1, une contrainte d'alphabet permis doit être ajoutée à la définition de type ASN.1 pour retirer les caractères TABULATION HORIZONTALE, NOUVELLE LIGNE et RETOUR CHARIOT. Une instruction de codage **WHITESPACE REPLACE** finale doit être allouée à la définition de type ASN.1. On doit utiliser la contrainte d'alphabet permis suivante ou son équivalent:

```
(FROM ({0, 0, 0, 32} .. {0, 16, 255, 255}))
```

12.3.2.3 Si la **valeur** de la facette **whiteSpace** est **collapse** *(réduire)* et si la définition de type ASN.1 correspondant à la **définition de type simple** est un type de chaîne de caractère restreint à l'ASN.1, une contrainte d'alphabet permis, comme spécifié au § 12.3.2.2 ainsi qu'une contrainte pattern interdisant les caractères ESPACE consécutifs, avant, après et multiples doivent toutes deux être ajoutées à la définition de type ASN.1. Une instruction de codage **WHITESPACE COLLAPSE** finale doit être allouée à la définition de type ASN.1. On doit utiliser la contrainte pattern suivante ou son équivalent:

```
(PATTERN "([^ ]([^ ]| [^ ])*)?")
```

12.4 La facette enumeration

12.4.1 Une facette **enumeration** appartenant à une **définition de type simple** avec une **variété** d'**atomic** qui est déduite par restriction (directement ou indirectement) à partir de **xsd:string** ne doit pas être mappée en contrainte ASN.1. Au lieu de cela, la facette doit être mappée en "Enumeration" du type énuméré de l'ASN.1 correspondant à la **définition de type simple** (voir le § 13.5) comme spécifié dans les trois paragraphes suivants.

12.4.1.1 Pour chaque membre de la **valeur** de la facette **enumeration**, un "EnumerationItem" qui est un "identifiant" doit être ajouté à l'"Enumeration" (conformément aux § 12.1.2, 12.2.1, 12.3.1 et 12.5.1).

12.4.1.2 Chaque "identifiant" doit être généré en appliquant le § 10.3 au membre correspondant de la **valeur** de la facette **enumeration**.

12.4.1.3 Les membres de la **valeur** de la facette **enumeration** doivent être mappés dans l'ordre lexicographique ascendant et tout membre faisant double emploi doit être éliminé.

12.4.2 Une facette **enumeration** appartenant à une **définition de type simple** avec une **variété** d'**atomic** qui est déduite par restriction (directement ou indirectement) à partir d'**xsd:integer** ne doit pas être mappée en contrainte ASN.1. Au lieu de cela, la facette doit être mappée en "Enumeration" du type énuméré de l'ASN.1 correspondant à la **définition de type simple** (voir le § 13.6) comme spécifié dans les trois paragraphes suivants.

12.4.2.1 Pour chaque membre de la **valeur** de la facette **enumeration**, un "EnumerationItem" qui est un "NamedNumber" doit être ajouté à l'"Enumeration" (conformément aux § 12.1.2, 12.2.1, 12.3.1 et 12.5.1).

12.4.2.2 L'"identifiant" dans chaque "NamedNumber" doit être généré en concaténant la chaîne de caractères "**int**" avec la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) du membre correspondant de la **valeur** de la facette **enumeration**. Le "SignedNumber" dans le "NamedNumber" doit être la notation de la valeur ASN.1 pour le membre (un nombre entier).

12.4.2.3 Les membres de la **valeur** de la facette **enumeration** doivent être mappés en ordre numérique ascendant et tout membre faisant double emploi doit être éliminé.

12.4.3 Toute autre facette **enumeration** doit être mappée en une contrainte ASN.1 qui sera soit une valeur unique soit une union de valeurs uniques correspondant aux membres de la **valeur** de l'**enumeration**.

NOTE – La facette **enumeration** s'applique à l'espace de valeur de la **définition de type de base**. Donc, pour une **enumeration** des datatypes incorporés de l'XSD **xsd:QName** ou **xsd:NOTATION**, la valeur du composant **uri** de la **SEQUENCE [USE-QNAME]** produite comme contrainte ASN.1 de valeur unique est déterminée, dans la représentation XML d'un schéma XSD, par les déclarations de namespace dont la portée inclut le **xsd:QName** ou la **xsd:NOTATION** et par le préfixe (s'il en est) du **xsd:QName** ou de la **xsd:NOTATION**.

EXEMPLE 1 – Ce qui suit représente une **définition de type simple** de niveau supérieur définie par l'usager qui est une restriction de **xsd:string** avec une facette **enumeration**:

```
<xsd:simpleType name="state">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="off"/>
                <xsd:enumeration value="on"/>
        </xsd:restriction>
</xsd:simpleType>
```

Elle est mappée dans l'allocation de type ASN.1:

```
        State ::= [NAME AS UNCAPITALIZED] ENUMERATED {off, on}
```

EXEMPLE 2 – Ce qui suit représente une **définition de type simple** de niveau supérieur définie par l'usager qui est une restriction de **xsd:integer** avec une facette **enumeration**:

```
<xsd:simpleType name="integer-0-5-10">
        <xsd:restriction base="xsd:integer">
                <xsd:enumeration value="0"/>
                <xsd:enumeration value="5"/>
                <xsd:enumeration value="10"/>
        </xsd:restriction>
</xsd:simpleType>
```

Elle est mappée dans l'allocation de type ASN.1:

```
        Integer-0-5-10 ::= [NAME AS UNCAPITALIZED] ENUMERATED {int0(0), int5(5), int10(10)}
```

EXEMPLE 3 – Ce qui suit représente une **définition de type simple** de niveau supérieur définie par l'usager qui est une restriction de **xsd:integer** avec une facette **minInclusive** et une facette **maxInclusive**:

```
<xsd:simpleType name="integer-1-10">
        <xsd:restriction base="xsd:integer">
```

```
                <xsd:minInclusive value="1"/>
                <xsd:maxInclusive value="10"/>
            </xsd:restriction>
        </xsd:simpleType>
```

Elle est mappée dans l'allocation de type ASN.1:

```
Integer-1-10 ::= [NAME AS UNCAPITALIZED] INTEGER(1..10)
```

EXEMPLE 4 – Ce qui suit représente une **définition de type simple** de niveau supérieur définie par l'usager qui est une restriction (avec une facette **minExclusive**) d'une autre **définition de type simple**, déduite par restriction de **xsd:integer** avec l'ajout d'une facette **minInclusive** et d'une facette **maxInclusive**:

```
<xsd:simpleType name="multiple-of-4">
    <xsd:restriction>
        <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
                <xsd:minInclusive value="1"/>
                <xsd:maxInclusive value="10"/>
            </xsd:restriction>
        </xsd:simpleType>
        <xsd:minExclusive value="5"/>
    </xsd:restriction>
</xsd:simpleType>
```

Elle est mappée dans l'allocation de type ASN.1:

```
Multiple-of-4 ::= [NAME AS UNCAPITALIZED] INTEGER(5<..10)
```

EXEMPLE 5 – Ce qui suit représente une **définition de type simple** de niveau supérieur définie par l'usager qui est une restriction (avec une facette **minLength** et **maxLength**) d'une autre **définition de type simple**, déduite par restriction à partir de **xsd:string** avec l'ajout d'une facette **enumeration**:

```
<xsd:simpleType name="color">
    <xsd:restriction>
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="white"/>
                <xsd:enumeration value="black"/>
                <xsd:enumeration value="red"/>
            </xsd:restriction>
        </xsd:simpleType>
        <xsd:minLength value="2"/>
        <xsd:maxLength value="4"/>
    </xsd:restriction>
</xsd:simpleType>
```

Elle est mappée dans l'allocation de type ASN.1:

```
Color ::= [NAME AS UNCAPITALIZED] ENUMERATED {red}
```

12.5 Autres facettes

12.5.1 Si une facette **totalDigits**, **fractionDigits**, **maxInclusive, maxExclusive**, **minExclusive** ou **minInclusive** appartient à une **définition de type simple** qui a aussi une facette **enumeration** en cours de mappage dans une "Enumeration" ASN.1 (voir les § 12.4.1 et 12.4.2), aucun "EnumerationItem" ne doit alors être inclus dans l'"Enumeration" pour les membres (s'il en est) de la **valeur** de la facette **enumeration** qui ne sont pas conformes à la facette **totalDigits**, **fractionDigits**, **maxInclusive, maxExclusive**, **minExclusive** ou **minInclusive**.

12.5.2 Si une facette **maxInclusive, maxExclusive**, **minExclusive** ou **minInclusive** appartient à une **définition de type simple** sans facette **enumeration** ou avec une facette **enumeration** qui n'est pas en cours de mappage en "Enumeration" ASN.1 (voir les § 12.4.1 et 12.4.2), un des deux paragraphes suivants s'applique alors:

12.5.2.1 Si la **définition de type simple** est déduite par restriction (directement ou indirectement) à partir d'un datatype de date ou d'heure incorporé en XSD (**xsd:date**, **xsd:dateTime**, **xsd:duration**, **xsd:gDay**, **xsd:gMonth**, **xsd:gYear**, **xsd:gYearMonth**, **xsd:gMonthDay** ou **xsd:time**), les facettes **maxInclusive**, **maxExclusive**, **minExclusive** et **minInclusive** de la **définition de type simple** doivent être mappées en contrainte ASN.1 définie par l'usager (voir le § 12.5.4).

12.5.2.2 Autrement, les facettes **maxInclusive**, **maxExclusive**, **minExclusive** et **minInclusive** de la **définition de type simple** doivent être mappées en contrainte ASN.1 de gamme de valeur ou de valeur unique conformément au Tableau 4.

Tableau 4 – Contraintes ASN.1 correspondant aux facettes maxInclusive, maxExclusive, minExclusive et minInclusive

Facette XSD	Contrainte ASN.1
maxInclusive=*ub*	(**MIN ..** *ub*)
maxExclusive=*ub*	(**MIN .. <** *ub*)
minExclusive=*lb*	(*lb* **<.. MAX**)
minInclusive=*lb*	(*lb* **.. MAX**)
minInclusive=*ub* **maxInclusive=***lb*	(*lb* **..** *ub*)
minInclusive=*v* **maxInclusive=***v*	(*v*)
minInclusive=*ub* **maxExclusive=***lb*	(*lb* **..<** *ub*)
minExclusive=*ub* **maxInclusive=***lb*	(*lb* **<..** *ub*)
minExclusive=*ub* **maxExclusive=***lb*	(*lb* **<..<** *ub*)

12.5.3 Si une facette **totalDigits** ou **fractionDigits** appartient à une **définition de type simple** sans facette **enumeration** ou avec une facette **enumeration** qui n'est pas en cours de mappage en "Enumeration" ASN.1 (voir les § 12.4.1 et 12.4.2), les facettes **totalDigits** et **fractionDigits** de la **définition de type simple** doivent être mappées en contrainte définie par l'usager (voir le § 12.5.4).

12.5.4 Lorsqu'une facette est mappée en contrainte ASN.1 définie par l'usager, il est recommandé que la facette et sa **valeur** apparaissent dans un commentaire ASN.1 dans la contrainte définie par l'usager. La forme précise de la contrainte définie par l'usager n'est pas spécifiée.

13 Mappage des définitions de type simple

13.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une allocation de type ASN.1 ou une définition de type ASN.1 correspondant à une **définition de type simple.**

13.2 Le présent paragraphe spécifie le mappage de **définition**s **de type simple** qui ne sont pas des datatypes incorporés en XSD. L'ensemble des datatypes incorporés en XSD est mappé dans le module prédéfini d'ASN.1 spécifié à l'Annexe A (le module **XSD**), qui doit être inclus dans les spécifications ASN.1 générées par le mappage.

13.3 Une **définition de type simple** de niveau supérieur définie par l'usager doit être mappée en allocation de type ASN.1. La "typereference" dans le "TypeAssignment" doit être générée en appliquant le § 10.3 au **nom** de la **définition de type simple** et le "Type" dans le "TypeAssignment" doit être une définition de type ASN.1 comme spécifié aux § 13.5 à 13.10.

13.4 Une **définition de type simple** anonyme doit être mappée en une définition de type ASN.1 comme spécifié aux § 13.5 à 13.10.

13.5 Pour une **définition de type simple** avec une **variété** d'**atomic** avec une facette **enumeration** déduite par restriction (directement ou indirectement) à partir de **xsd:string**, la définition de type ASN.1 doit être un type énuméré de l'ASN.1 dont l'"Enumeration" doit être généré comme spécifié au § 12.4.1.

13.6 Pour une **définition de type simple** avec une **variété** d'**atomic** avec une facette **enumeration** déduite par restriction (directement ou indirectement) à partir de **xsd:integer**, la définition de type ASN.1 doit être un type énuméré de l'ASN.1 dont l'"Enumeration" doit être généré comme spécifié au § 12.4.2. Une instruction de codage **USE-NUMBER** finale doit être allouée au type ASN.1 énuméré.

13.7 Pour toute autre **définition de type simple** (disons D) avec toute **variety** (variété) déduite par restriction (directement ou indirectement) à partir d'une **définition de type simple** de niveau supérieur définie par l'usager, la définition de type ASN.1 doit être générée en appliquant le § 23 à la **définition de type simple** de niveau supérieur définie par l'usager (disons B) de telle sorte que:

a) D est déduit par restriction (directement ou indirectement) de B; et

b) soit B est une **définition de type de base** de D, soit toutes les étapes intermédiaires de déduction de B à D sont des **définition**s **de type simple** anonymes.

Alors, pour chacune des **facettes** de D (s'il en est), une contrainte ASN.1 générée en appliquant le § 12 à la facette doit être ajoutée à la définition de type ASN.1.

13.8 Pour toute autre **définition de type simple** (disons D) avec une **variété** d'**atomic**, la définition de type ASN.1 doit être générée en appliquant le § 23 au datatype incorporé en XSD (disons B) de telle sorte que:

a) D est déduit par restriction (directement ou indirectement) de B; et

b) soit B est une **définition de type de base** de D, soit toutes les étapes intermédiaires de déduction de B à D sont des **définitions de type simple** anonymes.

Alors, pour chacune des **facettes** de D, une contrainte ASN.1 générée en appliquant le § 12 à la facette doit être ajoutée à la définition de type ASN.1.

13.9 Pour toute autre **définition de type simple** (disons D) avec une **variété** de **liste**, les trois paragraphes suivants s'appliquent.

13.9.1 La définition de type ASN.1 doit être d'un type séquence-de ASN.1 dont le composant doit être un "Type" généré en appliquant le § 23 à la **définition de type d'item**.

13.9.2 Pour chacune des **facettes** de D, une contrainte ASN.1 générée en appliquant le § 12 à la facette doit être ajoutée au type séquence-de ASN.1.

13.9.3 Une instruction de codage **`LIST`** finale doit être allouée au type séquence-de ASN.1.

EXEMPLE – Ce qui suit représente une **définition de type simple** de niveau supérieur définie par l'usager qui est une **liste** de **xsd:float:**

```
<xsd:simpleType name="list-of-float">
        <xsd:list itemType="xsd:float"/>
</xsd:simpleType>
```

Elle est mappée dans l'allocation de type ASN.1:

```
List-of-float ::= [LIST] [NAME AS UNCAPITALIZED] SEQUENCE OF XSD.Float
```

13.10 Pour toute autre **définition de type simple** (disons D) avec une **variété** d'**union**, les cinq paragraphes suivants s'appliquent.

13.10.1 La définition de type ASN.1 doit être un type de choix ASN.1 avec un choix pour chaque membre des **définitions de type de membre**.

13.10.2 Pour chaque membre des **définitions de type de membre**, l'"identifiant" dans le "NamedType" du choix correspondant doit être généré en appliquant le § 10.3 au **nom** du membre (si le membre est un datatype incorporé en XSD ou une **définition de type simple** de niveau supérieur définie par l'usager) ou à la chaîne de caractères "**`alt`**" (si le membre est une **définition de type simple** anonyme) et le "Type" dans le "NamedType" doit être la définition de type ASN.1 générée en appliquant le § 23 au membre des **définitions de type de membre**.

13.10.3 Pour chaque membre des **définitions de type de membre** qui est une **définition de type simple** anonyme, le "NamedType" correspondant doit avoir une instruction de codage **`NAME AS ""`** finale.

13.10.4 Pour chacune des **facettes** de D, une contrainte ASN.1 générée en appliquant le § 12 à la facette doit être ajoutée au type de choix ASN.1.

13.10.5 Une instruction de codage **`USE-UNION`** finale doit être allouée au type de choix ASN.1.

EXEMPLE – Ce qui suit représente une **définition de type simple** de niveau supérieur définie par l'usager qui est une **union** de deux **définitions de type simple** anonymes:

```
<xsd:simpleType name="decimalOrBinary">
        <xsd:union>
                <xsd:simpleType>
                        <xsd:restriction base="xsd:decimal"/>
                </xsd:simpleType>
                <xsd:simpleType>
                        <xsd:restriction base="xsd:float"/>
                </xsd:simpleType>
        </xsd:union>
</xsd:simpleType>
```

Ceci est mappé en l'allocation de type ASN.1:

```
DecimalOrBinary ::= [NAME AS UNCAPITALIZED] [USE-UNION] CHOICE {
            alt         [NAME AS ""] XSD.Decimal,
            alt-1       [NAME AS ""] XSD.Float }
```

14 Mappage des déclarations d'élément

14.1 Le présent paragraphe s'applique lorsqu'il est invoqué explicitement par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une allocation de type ASN.1 ou une définition de type ASN.1 correspondant à une **déclaration d'élément.**

NOTE – La présence d'une **contrainte de valeur** sur une **déclaration d'élément** affecte normalement le mappage. Cependant, le § 8.10 implique qu'une **déclaration d'élément** qui a une **contrainte de valeur** et dont la **définition de type** est **xsd:QName** ou **xsd:NOTATION** ou une restriction de ces datatypes incorporés en XSD soit mappée comme si elle n'avait pas de **contrainte de valeur**.

14.2 Une **déclaration d'élément** de niveau supérieur qui est **abstract** (*abstraite*) doit être ignorée.

14.3 Une **déclaration d'élément** de niveau supérieur qui n'est pas **abstraite** doit être mappée en allocation de type ASN.1. La "typereference" dans le "TypeAssignment" doit être générée en appliquant le § 10.3 au **nom** de la **déclaration d'élément** et le "Type" dans le "TypeAssignment" doit être une définition de type ASN.1 comme spécifié au § 14.5.

14.4 Une **déclaration d'élément** locale doit être mappée en une définition de type ASN.1 comme spécifié au § 14.5.

14.5 Un des deux paragraphes suivants (14.5.1 et 14.5.2) s'applique.

14.5.1 Si la **définition de type** de la **déclaration d'élément** est une **définition de type simple** anonyme ou une **définition de type complexe** ou un datatype incorporé en XSD (disons A), un des deux paragraphes suivants s'applique alors.

14.5.1.1 Si la **déclaration d'élément** n'est pas **nillable**, la définition de type ASN.1 doit alors être générée en appliquant le § 23 à A.

14.5.1.2 Si la **déclaration d'élément** est **nillable,** la définition de type ASN.1 doit alors être générée en appliquant le § 26 (si A est une **définition de type simple**) ou le § 27 (si A est une **définition de type complexe**) à A.

14.5.2 Si la **définition de type** de la **déclaration d'élément** est une **définition de type simple** de niveau supérieur définie par l'usager ou une **définition de type complexe** (disons T), un des quatre paragraphes suivants s'applique alors.

14.5.2.1 Si la **déclaration d'élément** n'est pas **nillable** et ne possède pas de **définition de type** substituable (voir le § 14.6), la définition de type ASN.1 doit alors être générée en appliquant le § 23 à T.

14.5.2.2 Si la **déclaration d'élément** est **nillable** et n'a pas de **définition de type** substituable (voir le § 14.6), la définition de type ASN.1 doit alors être générée en appliquant le § 26 (si T est une **définition de type simple**) ou le § 27 (si T est une **définition de type complexe**) à T.

14.5.2.3 Si la **déclaration d'élément** n'est pas **nillable** et a une **définition de type** substituable (voir le § 14.6), la définition de type ASN.1 doit alors être générée en appliquant le § 24 à T.

14.5.2.4 Si la **déclaration d'élément** est **nillable** et a une **définition de type** substituable (voir le § 14.6), la définition de type ASN.1 doit alors être générée en appliquant le § 25 à T.

14.6 La phrase "a une **définition de type** substituable", appliquée à une **déclaration d'élément**, signifie que la **définition de type** de la **déclaration d'élément** est une **définition de type simple** ou **définition de type complexe,** de niveau supérieur définie par l'usager, qui intervient comme **définition de type de base** d'une autre **définition de type simple** ou **définition de type complexe** de niveau supérieur.

15 Mappage des déclarations d'attribut

15.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une allocation de type ASN.1 ou une définition de type ASN.1 correspondant à une **déclaration d'attribut.**

15.2 Une **déclaration d'attribut** doit être mappée en une allocation de type ASN.1. La "typereference" dans le "TypeAssignment" doit être générée en appliquant le § 10.3 au **nom** de la **déclaration d'attribut** et le "Type" dans le "TypeAssignment" doit être une définition de type ASN.1 comme spécifié au § 15.4. Une instruction de codage **`ATTRIBUTE`** finale doit être attribuée à l'allocation de type ASN.1.

15.3 Une **déclaration d'attribut** local doit être mappée en définition de type ASN.1 comme spécifié au § 15.4.

15.4 La définition de type ASN.1 doit être générée en appliquant le § 23 à la **définition de type** de la **déclaration d'attribut.**

16 Valeurs de mappage de définitions de type simple

16.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une "Valeur" ASN.1 correspondant à une valeur dans l'espace de valeur d'une **définition de type simple.**

16.2 Etant donnée une valeur V dans l'espace de valeur d'une **définition de type simple** et:

a) la définition de type ASN.1 mappée à partir de cette **définition de type simple**; et

b) la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) de V,

V doit être mappé en une notation de valeur de base ASN.1 pour la valeur abstraite de la définition de type ASN.1 pour laquelle, en EXTENDED-XER, la représentation lexicale canonique est un codage valide de "ExtendedXMLValue".

17 Mappage des définitions de groupe modèle

17.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une allocation de type ASN.1 correspondant à une **définition de groupe modèle.**

17.2 Une **définition de groupe modèle** dont le **groupe modèle** a un **composeur** de **sequence** ou **choice** doit être mappée en une allocation de type ASN.1. Le "typereference" dans le "TypeAssignment" doit être généré en appliquant le § 10.3 au **nom** de la **définition de groupe modèle** et le "Type" dans le "TypeAssignment" doit être généré en appliquant le § 18 au **groupe modèle** de la **définition de groupe modèle.**

NOTE – Les **définitions de groupe modèle** dont le **groupe modèle** a un **composeur** de **all** (tout) ne sont pas mappées en ASN.1.

18 Mappage des groupes modèles

18.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une définition de type ASN.1 correspondant à un **groupe modèle.**

NOTE – Le présent paragraphe n'est pas invoqué pour tous les **groupes modèles**. Par exemple, un **groupe modèle** avec un **composeur** de **all** n'est pas mappé en ASN.1, mais ses **particules** sont mappées comme spécifié au § 20.9.

18.2 Un **groupe modèle** avec un **composeur** de **sequence** doit être mappé en type de séquence ASN.1. Pour chaque **particule** dans le **groupe modèle** dans l'ordre, un "NamedType" doit être généré en appliquant le § 19 à la **particule** et ce "NamedType" doit être ajouté au type de séquence comme un de ses composants. Une instruction de codage **`UNTAGGED`** finale doit être attribuée au type de séquence.

18.3 Un **groupe modèle** avec un **composeur** de **choix** doit être mappé en type de choix ASN.1. Pour chaque **particule** dans le **groupe modèle** dans l'ordre, un "NamedType" doit être généré en appliquant le § 19 à la **particule** et ce "NamedType" doit être ajouté au type de choix comme une de ses options. Une instruction de codage **`UNTAGGED`** finale doit être attribuée au type de choix.

19 Mappage des particules

19.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer un "NamedType" ASN.1 correspondant à une **particule.**

NOTE – Le présent paragraphe n'est pas invoqué pour toutes les **particule**s. Par exemple, la **particule** (la plus haute) du **type de contenu** d'une **définition de type complexe** est mappée d'une façon spéciale si son **terme** est un **groupe modèle** avec un **composeur** de **sequence** ou **tout** (voir le § 20.8).

19.2 Les trois paragraphes suivants définissent les termes qui sont utilisés dans le reste de ce paragraphe.

19.2.1 Si **min occurs** et **max occurs** d'une **particule** sont tous deux à un, la **particule** est appelée "particule à présence obligatoire".

19.2.2 Si **min occurs** est à zéro et **max occurs** est à un, alors:

a) si le mappage de la **particule** doit générer un composant d'un type de séquence ASN.1, la **particule** est appelée une "particule à présence facultative";

b) autrement, la **particule** est appelée une "particule facultative à occurrence unique".

19.2.3 Si **max occurs** est à deux ou plus, la **particule** est appelée une "particule à occurrence multiple".

19.3 Une "particule à présence obligatoire" ou une "particule à présence facultative" doit être mappée en "NamedType" comme spécifié dans les deux paragraphes suivants.

19.3.1 L'"identifiant" dans le "NamedType" doit être généré en appliquant le § 10.3 à la chaîne de caractères spécifiée au § 19.5 et le "Type" dans le "NamedType" doit être généré en appliquant le § 19.6 au **terme** de la **particule**.

19.3.2 Si la **particule** est une "particule à présence facultative", le "NamedType" doit être suivi du mot-clé **`OPTIONAL`**.

19.4 Une "particule facultative à occurrence unique" ou une "particule à occurrence multiple" doit être mappée en un "NamedType" comme spécifié dans les six paragraphes suivants.

19.4.1 L'"identifiant" dans le "NamedType" doit être généré en appliquant le § 10.3 à la chaîne de caractères obtenue en ajoutant le suffixe **`"-list"`** à la chaîne de caractères spécifiée au § 19.5. Le "Type" dans le "NamedType" doit être un type de séquence-de.

19.4.2 Si la **particule** est une "particule facultative à occurrence unique" ou une "particule à occurrence multiple", une contrainte de taille doit être ajoutée au type de séquence-de conformément au Tableau 5.

Tableau 5 – Contrainte de taille ASN.1 correspondant à min occurs et max occurs

min occurs et max occurs	**Contrainte de taille ASN.1**
min occurs = *n* **max occurs** = *n* $n \geq 2$	**`SIZE (`***n***`)`**
min occurs = *min* **max occurs** = *max* *max* > *min* et $max \geq 2$	**`SIZE (`***min* **`..`** *max***`)`**
min occurs = 0 max occurs = 1	**`SIZE (0 .. 1)`**
min occurs = *min* **max occurs** = **unbounded** $min \geq 1$	**`SIZE (`***min* **`.. MAX)`**
min occurs = 0 **max occurs = unbounded**	*pas de contrainte de taille*

19.4.3 Si le **terme** de la **particule** est une **déclaration d'élément**, le composant du type de séquence-de doit alors être un "NamedType". L'"identifiant" dans ce "NamedType" doit être généré en appliquant le § 10.3 au **nom** de la **déclaration d'élément** et le "Type" dans ce "NamedType" doit être généré en appliquant le § 19.6 au **terme** de la **particule**.

19.4.4 Si le **terme** de la **particule** est une **wildcard**, le composant du type de séquence-de doit alors être un "NamedType". L'"identifiant" dans ce "NamedType" doit être **`elem`** et le "Type" dans ce "NamedType" doit être généré en appliquant le § 19.6 au **terme** de la **particule**.

19.4.5 Si le **terme** de la **particule** est un **groupe modèle**, le composant du type de séquence-de doit alors être un "Type" et doit être généré en appliquant le § 19.6 au **terme** de la **particule**.

19.4.6 Une instruction de codage **`UNTAGGED`** finale doit être attribuée au type de séquence-de.

19.5 La chaîne de caractères utilisée dans la génération de l'"identifiant" dans le "NamedType" correspondant à la **particule** doit être:

a) si la **particule** est le **type de contenu** d'une **définition de type complexe**, le "content" de la chaîne de caractères;

b) si le **terme** de la **particule** est une **déclaration d'élément**, le **nom** de la **déclaration d'élément**;

c) si le **terme** de la **particule** est le **groupe modèle** d'une **définition de groupe modèle**, le **nom** de la **définition de groupe modèle**;

d) si le **terme** de la **particule** est un **groupe modèle** avec un **composeur** de **sequence** sans rapport avec une **définition de groupe modèle**, la chaîne de caractères "**`sequence`**";

e) si le **terme** de la **particule** est un **groupe modèle** avec un **composeur** de **choix** sans rapport avec une **définition de groupe modèle**, la chaîne de caractères "**`choice`**";

f) si le **terme** de la **particule** est une **wildcard**, la chaîne de caractères "**`elem`**".

19.6 Le "Type" dans le "NamedType" correspondant à la **particule** (voir le § 19.3) ou le "Type" dans le "NamedType" dans le "SequenceOfType" correspondant à la **particule** (voir le § 19.4) doit être:

a) si le **terme** de la **particule** est une **déclaration d'élément** de niveau supérieur qui n'est pas la tête d'un groupe de substitution d'élément, la définition de type ASN.1 (un "DefinedType") générée en appliquant le § 10.2 à l'allocation de type ASN.1 générée en appliquant le § 14 à la **déclaration d'élément**;

b) si le **terme** de la **particule** est une **déclaration d'élément** de niveau supérieur qui est la tête d'un groupe de substitution d'élément, la définition de type ASN.1 (un "DefinedType") générée en appliquant le § 10.2 à l'allocation de type ASN.1 générée en appliquant le § 31 à la **déclaration d'élément**;

c) si le **terme** de la **particule** est une **déclaration d'élément** locale, la définition de type ASN.1 générée en appliquant le § 14 à la **déclaration d'élément**;

d) si le **terme** de la **particule** est le **groupe modèle** d'une **définition de groupe modèle**, la définition de type ASN.1 (un "DefinedType") générée en appliquant le § 10.2 à l'allocation de type ASN.1 générée en appliquant le § 17 à la **définition de groupe modèle**;

e) si le **terme** de la **particule** est un **groupe modèle** sans rapport avec la **définition de groupe modèle**, la définition de type ASN.1 générée en appliquant le § 18 au **groupe modèle**;

f) si le **terme** de la **particule** est une **wildcard,** la définition de type ASN.1 générée en appliquant le § 21 à la **wildcard**.

20 Mappage des définitions de type complexe

20.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une allocation de type ASN.1 ou une définition de type ASN.1 correspondant à une **définition de type complexe.**

20.2 Une **définition de type complexe** de niveau supérieur doit être mappée en allocation de type ASN.1. Le "typereference" dans le "TypeAssignment" doit être généré en appliquant le § 10.3 au **nom** de la **définition de type complexe** et le "Type" dans le "TypeAssignment" doit être une définition de type ASN.1 comme spécifié aux § 20.4 à 20.11.

20.3 Une **définition de type complexe** anonyme doit être mappée en une définition de type ASN.1 comme spécifié aux § 20.4 à 20.11.

20.4 La définition de type ASN.1 doit être un type de séquence ASN.1. Zéro ou plusieurs composants doivent être ajoutés au type de séquence ASN.1 comme spécifié par les paragraphes suivants, dans l'ordre spécifié.

20.5 Si le **type de contenu** de la **définition de type complexe** est un modèle de contenu **mixte**, un composant doit alors être ajouté au type de séquence ASN.1. L'"identifiant" dans le "NamedType" de ce composant doit être **`embed-values`** et le "Type" dans le "NamedType" doit être un type de séquence-de dont le composant doit être un "Type" généré en appliquant le § 23 au datatype incorporé en XSD **xsd:string**. Une instruction de codage **`EMBED-VALUES`** finale doit être allouée au type de séquence ASN.1.

20.6 Si le **type de contenu** de la **définition de type complexe** est une **particule** dont le **terme** est un **groupe modèle** avec un **composeur** de **all**, un composant doit alors être ajouté au type de séquence ASN.1. L'"identifiant" dans le "NamedType" du composant doit être **`order`** et le "Type" dans le "NamedType" doit être un type de séquence-de dont le composant doit être un "EnumeratedType". Pour chaque **particule** du **groupe modèle** (dont le **terme** est toujours une **déclaration d'élément)**, un "EnumerationItem" qui est un "identifiant" identique à l'"identifiant" dans le "NamedType" correspondant à chaque **particule** doit être ajouté à l'"Enumeration" dans l'ordre. Une instruction de codage **`USE-ORDER`** finale doit être allouée au type de séquence ASN.1.

NOTE – Les "identifiants" dans les "NamedType" en cours de mappage à partir des **particules** sont générés (en appliquant le § 10.3) lorsque chaque composant est ajouté au type de séquence. Donc, même si le composant **`order`** est placé dans une position qui précède textuellement les positions des composants qui sont au sein du type de séquence ASN.1, la génération du composant **`order`** ne peut être complétée qu'après que toutes les **particules** ont été mappées en composants de séquence.

20.7 Si la **définition de type complexe** a des **attribute use**s, les composants générés en application du § 22 aux **attribute use**s doivent être ajoutés au type de séquence ASN.1 dans un ordre fondé sur le **namespace cible** et le **nom** de la **déclaration d'attribut** de chaque **attribute use**. Les **attribute use**s doivent d'abord être rangés par **namespace cible** de la **déclaration d'attribut** (avec le mot-clé **absent** précédant tous les noms de namespace triés en ordre lexicographique ascendant) puis ensuite par **nom** de la **déclaration d'attribut** au sein de chaque **namespace cible** (également en ordre lexicographique ascendant).

20.8 Si la **définition de type complexe** a une **attribute wildcard**, un composant généré à partir de l'**attribute wildcard** (voir le § 21.3) doit être alors ajouté au type de séquence ASN.1.

20.9 Si le **type de contenu** de la **définition de type complexe** est une **particule,** un des quatre paragraphes suivants s'applique alors.

20.9.1 Si le **terme** de la **particule** est un **groupe modèle** avec un **composeur** de **séquence** dont **min occurs** et **max occurs** sont tous deux à un, alors, pour chaque **particule** du **groupe modèle** dans l'ordre**,** un composant généré en appliquant le § 19 à la **particule** dans le **groupe modèle** doit être ajouté au type de séquence ASN.1.

20.9.2 Si le **terme** de la **particule** est un **groupe modèle** avec un **composeur** de **sequence** dont **min occurs** et **max occurs** ne sont pas tous deux à un, un composant généré en appliquant le § 19 à la **particule** dans le **type de contenu** doit être alors ajouté au type de séquence ASN.1.

20.9.3 Si le **terme** de la **particule** est un **groupe modèle** avec un **composeur** de **all**, alors, pour chaque **particule** du **groupe modèle** dans l'ordre, un composant généré en appliquant le § 19 à la **particule** du **groupe modèle** doit être ajouté au type de séquence ASN.1. Si la **particule** dans le **type de contenu** de la **définition de type complexe** a **min occurs** à zéro, chacune des **particules** du **groupe modèle** avec **min occurs** à un doit être mappée comme si elle avait **min occurs** à zéro.

20.9.4 Si le **terme** de la **particule** est un **groupe modèle** avec un **composeur** de **choix**, un composant généré en appliquant le § 19 à la **particule** dans le **type de contenu** doit être ajouté au type de séquence ASN.1.

20.10 Si le **type de contenu** de la **définition de type complexe** est une **définition de type simple**, un composant doit alors être ajouté au type de séquence ASN.1. L'"identifiant" dans le "NamedType" du composant doit être généré en appliquant le § 10.3 à la chaîne de caractères "**base**" et le "Type" dans le "NamedType" doit être la définition de type ASN.1 générée en appliquant le § 23 au **type de contenu.** Une instruction de codage **UNTAGGED** finale doit être allouée au composant.

20.11 Si le **type de contenu** de la **définition de type complexe** est **empty** (vide), aucun autre composant ne doit être ajouté au type de séquence ASN.1.

21 Mappage des wildcards

21.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une allocation de type ASN.1 ou une définition de type ASN.1 correspondant à une **définition de type simple.**

21.2 Une **wildcard** qui est le **terme** d'une **particule** doit être mappée en une définition de type ASN.1 générée en appliquant le § 23 au datatype incorporé en XSD **xsd:string**. Une instruction de codage **ANY-ELEMENT** finale doit être allouée à la définition de type ASN.1.

21.3 Une **wildcard** qui est la **wildcard attribut** d'un **type complexe** doit être mappée en un "NamedType". L'"identifiant" dans le "NamedType" doit être généré en appliquant le § 10.3 à la chaîne de caractères "**attr**" et le "Type" dans le "NamedType" doit être un type de séquence. Le composant du type séquence-de doit être un "Type" généré en appliquant le § 23 au datatype incorporé en XSD **xsd:string**. La contrainte définie par l'usager suivante doit être appliquée au type séquence-de:

```
(CONSTRAINED BY
     {/* Chaque item doit être conforme au "AnyAttributeFormat" spécifié dans
        la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */})
```

Une instruction de codage **ANY-ATTRIBUTES** finale doit être allouée au type séquence-de.

21.4 Si la **wildcard** a une **contrainte de namespace**, celle-ci doit être mappée en une "NameSpaceRestriction" dans l'instruction de codage **ANY-ELEMENT** ou **ANY-ATTRIBUTES**.

22 Mappage de attribute uses

22.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer un "NamedType" ASN.1 correspondant à un **attribute use** (utilisation d'attribut).

22.2 Un **attribute use** doit être mappé en un "NamedType".

22.3 L'"identifiant" dans le "NamedType" doit être généré en appliquant le § 10.3 au **nom** de la **déclaration d'attribut** de l'**attribute use** et le "Type" dans le "NamedType" doit être:

a) si l'**attribute use** a une **déclaration d'attribut** de niveau supérieur, la définition de type ASN.1 (un "DefinedType") générée en appliquant le § 10.2 à l'allocation de type ASN.1 générée en appliquant le § 15 à la **déclaration d'attribut**;

b) si l'**attribute use** a une **déclaration d'attribut** locale, la définition de type ASN.1 générée en appliquant le § 15 à la **déclaration d'attribut**.

22.4 Si ni l'**attribute use** ni sa **déclaration d'attribut** n'a de **contrainte de valeur**, le "NamedType" doit être suivi par le mot-clé **DEFAULT** et par une "Valeur" générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** de l'**attribute**

use (si l'**attribute use** a une **contrainte de valeur**) ou à la valeur dans la **contrainte de valeur** de sa **déclaration d'attribut** (autrement).

22.5 Si ni l'**attribute use** ni sa **déclaration d'attribut** n'a de **contrainte de valeur** qui soit une valeur **fixée**, une contrainte de valeur unique ASN.1 avec une "Valeur" identique à la "Valeur" suivant le mot-clé **DEFAULT** doit être ajoutée au "NamedType".

22.6 Si l'**attribute use** n'est pas **requis** et que ni l'**attribute use** ni sa **déclaration d'attribut** n'a de **contrainte de valeur**, le "NamedType" doit être suivi par le mot-clé **OPTIONAL** (*facultatif*).

22.7 Une instruction de codage **ATTRIBUTE** doit être allouée au "Type" dans le "NamedType".

23 Mappage des utilisations des définitions de type complexe et simple (cas général)

23.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une définition de type ASN.1 correspondant à l'utilisation d'une **définition de type simple** ou d'une **définition de type complexe**. Cela inclut leur utilisation comme **définition de type** de **déclaration**s **d'élément** qui n'ont pas une **définition de type** substituable (voir le § 14.6), ne sont pas **nillable**s et peuvent avoir ou ne pas avoir de **contrainte de valeur.**

23.2 Une utilisation de **définition de type simple** de niveau supérieur qui est un datatype incorporé en XSD doit être mappée comme spécifié au § 11.

23.3 Une utilisation d'une **définition de type simple** de niveau supérieur définie par l'usager doit être mappée en une définition de type ASN.1 (un "DefinedType") générée en appliquant le § 10.2 à l'allocation de type ASN.1 générée en appliquant le § 13 à la **définition de type simple**.

23.4 Une utilisation d'une **définition de type complexe** de niveau supérieur doit être mappée en une définition de type ASN.1 (un "DefinedType") générée en appliquant le § 10.2 à l'allocation de type ASN.1 générée en appliquant le § 20 à la **définition de type complexe**.

23.5 Une utilisation d'une **définition de type simple** anonyme n'est pas distinguée de la **définition de type simple** elle-même et doit être mappée comme spécifié au § 13 pour la **définition de type simple**.

23.6 Une utilisation d'une **définition de type complexe** anonyme n'est pas distinguée de la **définition de type complexe** elle-même et doit être mappée comme spécifié au § 20 pour la **définition de type complexe**.

23.7 Si une **définition de type simple** ou une **définition de type complexe** est utilisée comme **définition de type** d'une **déclaration d'élément** avec une **contrainte de valeur**, une instruction de codage **DEFAULT-FOR-EMPTY** finale doit alors être allouée à la définition de type ASN.1 et un des trois paragraphes suivants s'applique.

23.7.1 Pour une **définition de type simple,** la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de la **définition de type simple**.

23.7.2 Pour une **définition de type complexe** dont le **type de contenu** est une **définition de type simple**, la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de la **définition de type simple**.

23.7.3 Pour une **définition de type complexe** avec un type de contenu **mixte**, la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de **xsd:string** avec **whiteSpace preserve**.

23.8 Si une **définition de type simple** ou une **définition de type complexe** est utilisée comme **définition de type** d'une **déclaration d'élément** avec une **contrainte de valeur** qui est une valeur **fixe**, un des trois paragraphes suivants s'applique alors.

23.8.1 Pour une **définition de type simple,** une contrainte ASN.1 de valeur unique avec une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être ajoutée à la définition ASN.1.

23.8.2 Pour une **définition de type complexe** dont le **type de contenu** est une **définition de type simple**, une contrainte ASN.1 de sous-type interne doit être ajoutée à la définition ASN.1 et doit appliquer au composant de **base** une contrainte de valeur unique avec une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale.

23.8.3 Pour une **définition de type complexe** avec un type de contenu **mixte**, une contrainte ASN.1 de sous-type interne doit être ajoutée à la définition ASN.1 et doit appliquer:

a) au composant **embed-values**, une contrainte ASN.1 de valeur unique avec une "Valeur" consistant en une seule occurrence d'une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale;

b) à chaque composant qui est **OPTIONAL** et n'a pas d'instruction de codage **ATTRIBUTE** finale, le mot-clé **ABSENT**; et

c) à chaque composant dont le type est un type séquence-de, une contrainte **SIZE(0)**.

24 Mappage des utilisations spéciales des définitions de type simple et complexe (substituable)

24.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une définition de type ASN.1 correspondant à une **définition de type simple** ou une **définition de type complexe** utilisée comme **définition de type** de **déclaration**s **d'élément** qui ont une **définition de type** substituable (voir le § 14.6), ne sont pas **nillable**s et peuvent avoir ou pas une **contrainte de valeur**.

24.2 Une utilisation d'une **définition de type simple** (disons STD) ou une **définition de type complexe** (disons CTD) doit être mappée en un type de choix ASN.1.

24.3 Une option doit être ajoutée au type de choix ASN.1 pour STD ou CTD elle-même et une option doit être ajoutée pour chaque **définition de type simple** et **définition de type complexe** de niveau supérieur définie par l'usager dans le schéma XSD de source qui est déduit par restriction ou extension (directement ou indirectement) à partir de STD ou CTD.

24.4 Pour chaque option, l'"identifiant" dans le "NamedType" doit être généré en appliquant le § 10.3 au **nom** de la **définition de type simple** ou **définition de type complexe** correspondant à l'option et le "Type" dans le "NamedType" doit être la définition de type ASN.1 générée en appliquant le § 23 à la **définition de type simple** ou **définition de type complexe**.

24.5 La première option ajoutée au type de choix doit être celle correspondant à STD ou CTD elle-même. Les options suivantes doivent être ajoutées au type de choix dans un ordre fondé sur le **namespace** et **nom cible** des **définition**s **de type simple** et **définition**s **de type complexe**. Les définitions de type doivent d'abord être ordonnées par **namespace cible** (avec le namespace **absent** précédant tous les noms de namespace triés dans l'ordre lexicographique ascendant) puis par **nom** (aussi en ordre lexicographique ascendant) au sein de chaque **namespace cible**.

24.6 Une instruction de codage **USE-TYPE** finale doit être allouée au type de choix ASN.1.

24.7 S'il y a une **contrainte de valeur**, une instruction de codage **DEFAULT-FOR-EMPTY** finale doit être allouée à chaque option du type de choix ASN.1. Un des trois paragraphes suivants s'applique.

24.7.1 Si l'option correspond à une **définition de type simple,** la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de la **définition de type simple**.

24.7.2 Si l'option correspond à une **définition de type complexe** dont le **type de contenu** est une **définition de type simple**, la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de la **définition de type simple**.

24.7.3 Si l'option correspond à une **définition de type complexe** avec un type de contenu **mixte**, la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de **xsd:string** avec **whiteSpace preserve**.

24.8 S'il y a une **contrainte de valeur** qui est une valeur **fixe**, une contrainte de sous-type ASN.1 interne doit être ajoutée au type de choix ASN.1. Un des trois paragraphes suivants s'applique.

24.8.1 Si l'option correspond à une **définition de type simple,** la contrainte de sous-type interne doit appliquer à l'option une contrainte de valeur unique ASN.1 avec une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale.

24.8.2 Si l'option correspond à une **définition de type complexe** dont le **type de contenu** est une **définition de type simple**, la contrainte de sous-type interne doit appliquer à l'option une autre contrainte de sous-type ASN.1 interne qui applique au composant de **base** une contrainte de valeur unique avec une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale.

24.8.3 Si l'option correspond à une **définition de type complexe** avec un type de contenu **mixte**, la contrainte de sous-type interne doit appliquer à l'option une autre contrainte de sous-type ASN.1 interne qui applique:

a) au composant **embed-values**, une contrainte de valeur unique ASN.1 avec une "Valeur" consistant en une seule occurrence d'une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale;

b) à chaque composant qui est **OPTIONAL** et n'a pas d'instruction de codage **ATTRIBUTE** finale, le mot-clé **ABSENT**; et

c) à chaque composant dont le type est un type séquence-de, une contrainte **SIZE(0)**.

25 Mappage des utilisations spéciales des définitions de type simple et complexe (substituable, nillable)

25.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une définition de type ASN.1 correspondant à une **définition de type simple** ou une **définition de type complexe** utilisée comme **définition de type** de **déclaration**s **d'élément** qui ont une **définition de type** substituable (voir le § 14.6), sont **nillable**s et peuvent avoir ou ne pas avoir une **contrainte de valeur**.

25.2 Une utilisation d'une **définition de type simple** (disons STD) ou une **définition de type complexe** (disons CTD) doit être mappée en un type de choix ASN.1.

25.3 Une option doit être ajoutée au type de choix ASN.1 pour STD ou CTD elle-même et une option doit être ajoutée pour chaque **définition de type simple** et chaque **définition de type complexe** de niveau supérieur définie par l'usager dans le schéma XSD source qui est déduit par restriction ou extension (directement ou indirectement) à partir de STD ou de CTD.

25.4 Pour chaque option, l'"identifiant" dans le "NamedType" doit être généré en appliquant le § 10.3 au **nom** de la **définition de type simple** ou de la **définition de type complexe** correspondant à l'option et le "Type" dans le "NamedType" doit être la définition de type ASN.1 (un "DefinedType") généré en appliquant le § 10.2 à l'allocation de type ASN.1 générée en appliquant le § 30 à la **définition de type simple** ou à la **définition de type complexe**.

25.5 La première option ajoutée au type de choix doit être celle correspondant à STD ou CTD elle-même. Les options suivantes doivent être ajoutées au type de choix dans un ordre fondé sur le **namespace cible** et le **nom** des **définition**s **de type simple** et des **définition**s **de type complexe**. Les définitions de type doivent d'abord être ordonnées par **namespace cible** (avec le namespace **absent** précédant tous les noms de namespace triés en ordre lexicographique ascendant) et ensuite par **nom** (aussi en ordre lexicographique ascendant) au sein de chaque **namespace cible**.

25.6 Une instruction de codage **USE-TYPE** finale doit être allouée au type de choix ASN.1.

25.7 S'il y a une **contrainte de valeur**, une instruction de codage **DEFAULT-FOR-EMPTY** finale doit être allouée à chaque option du type de choix ASN.1. On applique un des trois paragraphes suivants.

25.7.1 Si l'option correspond à une **définition de type simple,** la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de la **définition de type simple**.

25.7.2 Si l'option correspond à une **définition de type complexe** dont le **type de contenu** est une **définition de type simple**, la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de la **définition de type simple**.

25.7.3 Si l'option correspond à une **définition de type complexe** avec un type de contenu **mixte**, la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de **xsd:string** avec **whiteSpace preserve**.

25.8 S'il y a une **contrainte de valeur** qui a une valeur **fixée**, une contrainte de sous-type ASN.1 interne doit être ajoutée au type de choix ASN.1. On applique un des trois paragraphes suivants.

25.8.1 Si l'option correspond à une **définition de type simple,** la contrainte de sous-type interne doit appliquer à l'option (qui est un type de séquence ASN.1 avec une instruction de codage **USE-NIL** finale) une autre contrainte de sous-type ASN.1 interne qui à son tour doit appliquer au composant **content** (*contenu*) le mot-clé **PRESENT** et une contrainte de valeur unique ASN.1 avec une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale.

25.8.2 Si l'option correspond à une **définition de type complexe** dont le **type de contenu** est une **définition de type simple**, la contrainte de sous-type interne doit appliquer à l'option (qui est un type de séquence ASN.1 avec une instruction de codage **USE-NIL** finale) une autre contrainte de sous-type ASN.1 interne qui applique au composant **content** le mot-

clé **PRESENT** et une contrainte de valeur unique ASN.1 avec une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale.

25.8.3 Si l'option correspond à une **définition de type complexe** avec un type de contenu **mixte**, la contrainte de sous-type interne doit appliquer à l'option (qui est un type de séquence ASN.1 avec une instruction de codage **USE-NIL** finale) une autre contrainte de sous-type ASN.1 interne qui applique:

a) au composant **embed-values**, une contrainte de valeur unique ASN.1 avec une "Valeur" consistant en une seule occurrence d'une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale;

b) au composant **content** (qui est un type de séquence ASN.1), le mot-clé **PRESENT** et une autre contrainte de sous-type interne qui applique le mot-clé **ABSENT** à chacun de ses composants qui est **OPTIONAL** et une contrainte **SIZE(0)** à chacun de ses composants dont le type est un type séquence-de;

c) à chaque composant qui est **OPTIONAL** et n'a pas d'instruction de codage **ATTRIBUTE** finale, le mot-clé **ABSENT**; et

d) à chaque composant dont le type est un type séquence-de, une contrainte **SIZE(0)**.

26 Mappage des utilisations spéciales des définitions de type simple (nillable)

26.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une définition de type ASN.1 correspondant à une **définition de type simple** utilisée comme la **définition de type** des **déclaration**s **d'élément** qui n'ont pas une **définition de type** substituable (voir le § 14.6), sont **nillable**s et peuvent avoir ou ne pas avoir une **contrainte de valeur**.

26.2 Une utilisation d'une **définition de type simple** doit être mappée en un type de séquence ASN.1 avec un composant **OPTIONAL**.

26.3 L'"identifiant" dans le "NamedType" du composant doit être **content** et le "Type" dans le "NamedType" doit être la définition de type ASN.1 générée en appliquant le § 23 à la **définition de type simple**.

26.4 Une instruction de codage **USE-NIL** finale doit être allouée au type de séquence ASN.1.

26.5 S'il y a une **contrainte de valeur**, une instruction de codage **DEFAULT-FOR-EMPTY** finale doit être allouée au type de séquence ASN.1. La "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur**.

26.6 S'il y a une **contrainte de valeur** qui est une valeur **fixée**, une contrainte de sous-type ASN.1 interne doit être ajoutée au type de séquence ASN.1. La contrainte de sous-type interne doit appliquer au composant **content** une contrainte de valeur unique ASN.1 avec une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale. La contrainte de sous-type interne doit aussi appliquer le mot-clé **PRESENT** au composant **content**.

27 Mappage des utilisations spéciales des définitions de type complexe (nillable)

27.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une définition de type ASN.1 correspondant à une **définition de type complexe** utilisée comme la **définition de type** des **déclaration**s **d'élément** qui n'ont pas une **définition de type** substituable (voir le § 14.6), sont **nillable**s et peuvent avoir ou ne pas avoir une **contrainte de valeur**.

27.2 Une utilisation d'une **définition de type complexe** doit être mappée en un type de séquence ASN.1. Un ou plusieurs composants doivent être ajoutés au type de séquence ASN.1 comme spécifié dans les paragraphes suivants, dans l'ordre spécifié.

27.3 Si le **type de contenu** de la **définition de type complexe** est un modèle de contenu **mixte**, un composant **embed-values** doit être ajouté au type de séquence ASN.1 comme spécifié au § 20.5.

27.4 Si le **type de contenu** de la **définition de type complexe** est **particule** dont le **terme** est un **groupe modèle** avec un **composeur** de **all**, un composant **order** doit alors être ajouté au type de séquence ASN.1 comme spécifié au § 20.6.

27.5 Si la **définition de type complexe** a **attribute use**s, les composants mappés à partir des **attribute use**s doivent être ajoutés au type de séquence ASN.1 comme spécifié au § 20.7.

27.6 Si la **définition de type complexe** a un **attribute wildcard**, un composant généré à partir de **attribute wildcard** doit alors être ajouté au type de séquence ASN.1 comme spécifié au § 20.8.

27.7 Si le **type de contenu** de la **définition de type complexe** est une **particule,** on applique alors un des deux paragraphes suivants.

27.7.1 Si le **terme** de la **particule** est un **groupe modèle** avec un **composeur** de **sequence** ou de **choice**, un composant **OPTIONAL** doit alors être ajouté au type de séquence ASN.1. L'"identifiant" dans le "NamedType" du composant doit être généré en appliquant le § 10.3 à la chaîne de caractères "**content**" et le "Type" dans le "NamedType" doit être un type de séquence ASN.1 avec un seul composant, qui doit être généré en appliquant le § 19 à la **particule** dans le **type de contenu**.

27.7.2 Si le **terme** de la **particule** est un **groupe modèle** avec un **composeur** de **all**, un composant **OPTIONAL** doit alors être ajouté au type de séquence ASN.1. L'"identifiant" dans le "NamedType" du composant doit être généré en appliquant le § 10.3 à la chaîne de caractères "**content**" et le "Type" dans le "NamedType" doit être un type de séquence ASN.1. Pour chaque **particule** du **groupe modèle** dans l'ordre, un composant généré en appliquant le § 19 à la **particule** du **groupe modèle** doit être ajouté au type de séquence ASN.1 interne. Si la **particule** dans le **type de contenu** de la **définition de type complexe** a **min occurs** à zéro, chacune des **particules** du **groupe modèle** avec **min occurs** à un doit être mappée comme si elle avait **min occurs** à zéro.

27.8 Si le **type de contenu** de la **définition de type complexe** est une **définition de type simple**, un composant **OPTIONAL** doit être ajouté au type de séquence ASN.1. L'"identifiant" dans le "NamedType" du composant doit être généré en appliquant le § 10.3 à la chaîne de caractères "**content**" et le "Type" dans le "NamedType" doit être la définition de type ASN.1 générée en appliquant le § 23 au **type de contenu.**

27.9 Si le **type de contenu** de la **définition de type complexe** est **empty** (*vide*), aucun autre composant ne doit être ajouté au type de séquence ASN.1.

27.10 Une instruction de codage **USE-NIL** finale doit être allouée au type de séquence ASN.1.

27.11 S'il y a une **contrainte de valeur**, une instruction de codage **DEFAULT-FOR-EMPTY** finale doit être allouée au type de séquence ASN.1. On applique un des deux paragraphes suivants.

27.11.1 Si le **type de contenu** de la **définition de type complexe** est une **définition de type simple**, la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de la **définition de type simple**.

27.11.2 Si le **type de contenu** de la **définition de type complexe** est un type de contenu **mixte**, la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale doit être générée en appliquant le § 16 à la valeur dans la **contrainte de valeur** considérée comme une valeur dans l'espace de valeur de **xsd:string** avec **whiteSpace preserve**.

27.12 S'il y a une **contrainte de valeur** qui est une valeur **fixe**, une contrainte de sous-type ASN.1 interne doit être ajoutée au type de séquence ASN.1. La contrainte de sous-type interne doit appliquer le mot-clé **PRESENT** au composant **content**. On applique un des deux paragraphes suivants.

27.12.1 Si le **type de contenu** de la **définition de type complexe** est une **définition de type simple**, la contrainte de sous-type interne doit appliquer au composant **content** une contrainte de valeur unique ASN.1 avec une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale.

27.12.2 Si le **type de contenu** de la **définition de type complexe** est un type de contenu **mixte**, la contrainte de sous-type interne doit appliquer:

a) au composant **embed-values**, une contrainte de valeur unique ASN.1 avec une "Valeur" consistant en une seule occurrence d'une "Valeur" identique à la "Valeur" dans l'instruction de codage **DEFAULT-FOR-EMPTY** finale;

b) à chaque composant du composant **content** (un type de séquence ASN.1) qui est **OPTIONAL**, le mot-clé **ABSENT**;

c) à chaque composant du composant **content** (un type de séquence ASN.1) dont le type est un type séquence-de, une contrainte **SIZE(0)**.

28 Mappage des utilisations spéciales des déclarations d'élément (en-tête de groupe de substitution d'élément)

28.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une définition de type ASN.1 correspondant à une **déclaration d'élément** de niveau supérieur qui est l'en-tête d'un groupe de substitution d'élément et est utilisée comme le **terme** des **particule**s.

28.2 Une utilisation d'une **déclaration d'élément** de niveau supérieur doit être mappée en type de choix ASN.1.

28.3 Une option doit être ajoutée au type de choix ASN.1 pour la **déclaration d'élément** de niveau supérieur elle-même (disons H) et une option doit être ajoutée pour chaque **déclaration d'élément** de niveau supérieur dans le schéma XSD source qui n'est pas **abstrait** et dont **l'affiliation de groupe de substitution** est H.

28.4 Pour chaque option, l'"identifiant" dans le "NamedType" doit être généré en appliquant le § 10.3 au **nom** de la **déclaration d'élément** de niveau supérieur correspondant à l'option et le "Type" dans le "NamedType" doit être la définition de type ASN.1 (un "DefinedType") généré en appliquant le § 10.2 à l'allocation de type ASN.1 générée en appliquant le § 14 à la **déclaration d'élément** de niveau supérieur.

NOTE – En XSD, l'appartenance à un groupe de substitution est transitive, c'est-à-dire que les membres d'un groupe de substitution ESG1 dont l'en-tête est un membre d'un autre groupe de substitution ESG2 sont tous aussi membres de ESG2.

28.5 Des options doivent être ajoutées au type de choix dans un ordre fondé sur le **namespace cible** et le **nom** des **déclaration**s **d'élément** de niveau supérieur. Les **déclaration**s **d'élément** doivent d'abord être ordonnées par **namespace cible** (avec le namespace **absent** précédant tous les noms de namespace triés en ordre lexicographique ascendant) et ensuite par **nom** (aussi en ordre lexicographique ascendant) au sein de chaque **namespace cible**.

NOTE – La **déclaration d'élément** qui est l'en-tête du groupe de substitution d'élément est ordonnée avec les autres **déclaration**s **d'élément** qui appartiennent au groupe de substitution d'élément.

28.6 Une instruction de codage **`UNTAGGED`** finale doit être allouée au type de choix.

29 Génération d'allocations de type ASN.1 spéciales pour les déclarations d'élément

29.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une allocation de type ASN.1 correspondant à une **définition de type simple** ou une **définition de type complexe** de niveau supérieur définie par l'usager utilisée comme **définition de type** des **déclaration**s **d'élément** qui ont une **définition de type** substituable (voir le § 14.6) ou sont **nillables**.

29.2 Le présent paragraphe est invoqué par d'autres paragraphes pour une combinaison donnée de:

a) **définition de type simple** ou de **définition de type complexe**;

b) au cas où les **déclaration**s **d'élément** ont une **définition de type** substituable (voir le § 14.6);

c) au cas où la **déclaration d'élément** est **nillable**; et

d) au cas où la **déclaration d'élément** a une **contrainte de valeur** et l'aspect et la valeur de la **contrainte de valeur**;

et génère une allocation de type ASN.1 (appelée "allocation spéciale de type ASN.1 (pour les déclarations d'élément)") pour une combinaison des items ci-dessus.

29.3 Une allocation spéciale de type ASN.1 et une seule doit être générée pour chaque différente combinaison des items ci-dessus qui surviennent réellement dans une ou plusieurs invocations de ce paragraphe à travers le mappage d'un schéma XSD de source.

NOTE – Par exemple, si deux **déclaration**s **d'élément** ou plus, dans un grand schéma XSD ont des **définition**s **de type** identique, sont toutes deux **nillable**s et ont toutes deux une **contrainte de valeur** qui est une valeur par **défaut** et qui est la même valeur, une allocation spéciale de type ASN.1 unique est alors générée. Le nom de référence de type de cette allocation de type surviendra dans le "Type" dans les "TypeAssignment" correspondant aux deux **déclaration**s **d'élément**.

29.4 Le terme "allocation associée de type ASN.1" désigne l'allocation de type ASN.1 en cours de mappage à partir de la **définition de type simple** ou de la **définition de type complexe** qui est la **définition de type** de la **déclaration d'élément** pour laquelle une allocation spéciale de type ASN.1 est générée, en appliquant respectivement le § 13 ou le § 20.

NOTE – Toute allocation spéciale de type ASN.1 a une allocation associée de type ASN.1, car ce paragraphe ne s'applique que lorsque la **définition de type** d'une **déclaration d'élément** est une **définition de type simple** ou une **définition de type complexe** de niveau supérieur définie par l'usager. Toutes les **définition**s **de type simple** et les **définition**s **de type complexe** de cette sorte sont mappées en allocations de type ASN.1.

29.5 Pour une **déclaration d'élément** donnée, le "typereference" dans le "TypeAssignment" pour une allocation spéciale de type ASN.1 doit être construite en ajoutant un suffixe au nom de référence de type de l'allocation associée de type ASN.1 et en appliquant le § 10.3 à la chaîne de caractères résultante et le "Type" dans le "TypeAssignment" doit être la définition de type ASN.1 générée en appliquant le § 24 ou le § 25 à la **définition de type simple** ou à la **définition de type complexe** qui est la **définition de type** de la **déclaration d'élément**. On applique un des alinéas suivants:

a) si la **déclaration d'élément** n'est pas **nillable**, a une **définition de type** substituable et n'a pas de **contrainte de valeur**, le suffixe doit être "**`-derivations`**" et on doit appliquer le § 24;

b) si la **déclaration d'élément** est **nillable**, a une **définition de type** substituable, n'a pas de **contrainte de valeur**, le suffixe doit être "**`-deriv-nillable`**" et on doit appliquer le § 25;

c) si la **déclaration d'élément** n'est pas **nillable**, a une **définition de type** substituable et a une **contrainte de valeur** qui est une valeur par **défaut**, le suffixe doit être "`-deriv-default-`" suivi par la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) de la valeur dans la **contrainte de valeur** et on doit appliquer le § 24;

d) si la **déclaration d'élément** n'est pas **nillable**, a une **définition de type** substituable et a une **contrainte de valeur** qui est une valeur **fixée**, le suffixe doit être "`-deriv-fixed-`" suivi par la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) de la valeur dans la **contrainte de valeur** et on doit appliquer le § 24;

e) si la **déclaration d'élément** est **nillable** et a une **définition de type** substituable et a une **contrainte de valeur** qui est une valeur par **défaut**, le suffixe doit être "`-deriv-nillable-default-`" suivi par la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) de la valeur dans la **contrainte de valeur** et on doit appliquer le § 25;

f) si la **déclaration d'élément** est **nillable**, a une **définition de type** substituable et a une **contrainte de valeur** qui est une valeur **fixée**, le suffixe doit être "`-deriv-nillable-fixed-`" suivi par la représentation lexicale canonique (voir la Partie 2, § 2.3.1 du schéma XML du W3C) de la valeur dans la **contrainte de valeur** et on doit appliquer le § 25.

30 Génération d'allocations spéciales de type ASN.1 pour les définitions de type

30.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une allocation de type ASN.1 correspondant à une **définition de type simple** ou une **définition de type complexe** de niveau supérieur définie par l'usager qui appartient à la hiérarchie de déduction de la **définition de type** des **déclaration**s **d'élément** qui ont une **définition de type** substituable (voir le § 14.6) et sont **nillable**s.

30.2 Le présent paragraphe est invoqué par les autres paragraphes pour une **définition de type simple** ou une **définition de type complexe** donnée et génère une allocation de type ASN.1 (appelée une "allocation spéciale de type ASN.1 (pour une définition de type)").

30.3 Une allocation spéciale de type ASN.1 et une seule doit être générée pour chaque **définition de type simple** ou **définition de type complexe** qui survient réellement dans une ou plusieurs invocations de ce paragraphe lors du mappage d'un schéma XSD de source.

30.4 Le terme "allocation associée de type ASN.1" désigne l'allocation de type ASN.1 en cours de mappage à partir de la **définition de type simple** ou de la **définition de type complexe** en appliquant respectivement le § 13 ou le § 20.

30.5 Le "typereference" dans le "TypeAssignment" pour une allocation spéciale de type ASN.1 doit être construit en ajoutant le suffixe "`-nillable`" au nom de référence de type de l'allocation associée de type ASN.1 et en appliquant le § 10.3 à la chaîne de caractères résultante et le "Type" dans le "TypeAssignment" doit être la définition de type ASN.1 générée en appliquant respectivement le § 26 ou le § 27 à la **définition de type simple** ou à la **définition de type complexe**.

31 Génération des allocations spéciales de type ASN.1 pour les groupes de substitution d'élément

31.1 Le présent paragraphe s'applique lorsqu'il est explicitement invoqué par d'autres paragraphes de la présente Recommandation | Norme internationale pour générer une allocation de type ASN.1 correspondant à une **particule** dont le **terme** est une **déclaration d'élément** de niveau supérieur qui est l'en-tête d'un groupe de substitution d'élément.

31.2 Le présent paragraphe est invoqué par d'autres paragraphes pour une **déclaration d'élément** de niveau supérieur qui est l'en-tête d'un groupe de substitution d'élément et génère une allocation de type ASN.1 (appelée "allocation spéciale de type ASN.1 (pour un groupe de substitution d'élément)").

31.3 Une allocation spéciale de type ASN.1 et une seule doit être générée pour chaque **déclaration d'élément** de niveau supérieur qui survient réellement dans une ou plusieurs invocations du présent paragraphe lors du mappage d'un schéma XSD de source.

31.4 Le terme "allocation associée de type ASN.1" désigne l'allocation de type ASN.1 en cours de mappage à partir de la **déclaration d'élément** de niveau supérieur en appliquant le § 14.

31.5 Le "typereference" dans le "TypeAssignment" pour une allocation spéciale de type ASN.1 doit être construit en ajoutant le suffixe "`-group`" au nom de référence de type de l'allocation associée de type ASN.1 et en appliquant le § 10.3 à la chaîne de caractères résultante et le "Type" dans le "TypeAssignment" doit être la définition de type ASN.1 générée en appliquant le § 28 à la **déclaration d'élément** de niveau supérieur.

Annexe A

Définitions de type ASN.1 correspondant aux datatypes incorporés en XSD

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

A.1 La présente annexe spécifie un module qui définit les types ASN.1 qui correspondent aux datatypes incorporés en XSD et qui sont utilisés pour le mappage en ASN.1 à partir du schéma XML du W3C.

A.2 Le schéma XML du W3C définit de nombreux datatypes incorporés et de temps pour représenter les durées, les instants ou les instants récurrents. Bien qu'ils soient tous déduits de la norme ISO 8601, il y a quelques extensions et restrictions. Les datatypes de date et heure incorporés en XSD sont mappés en **VisibleString** avec une contrainte définie par l'usager qui fait référence à la section XSD applicable. Une contrainte d'alphabet autorisé est ajoutée pour procurer un codage plus efficace avec les règles de codage compact de la notation ASN.1 (PER, *packed encoding rules*), dans la mesure où les contraintes définies par l'usager ne sont pas visibles du point de vue des PER (et par conséquent ne sont pas utilisées pour optimiser les codages).

A.3 Le module **XSD** est:

```
XSD {joint-iso-itu-t asn1(1) specification(0) modules(0) xsd-module(2)}
DEFINITIONS
AUTOMATIC TAGS ::=
BEGIN

/* xsd:anySimpleType */

AnySimpleType ::= XMLCompatibleString

/* xsd:anyType */

AnyType ::= SEQUENCE {
      embed-values SEQUENCE OF String,
      attr SEQUENCE
        (CONSTRAINED BY {
          /* chaque item doit être conforme au "AnyAttributeFormat" spécifié
           dans la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */ } ) OF String,
      elem-list SEQUENCE OF elem String
        (CONSTRAINED BY {
           /* doit être conforme au "AnyElementFormat" spécifié
              dans la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 19 */ } ) }
      (CONSTRAINED BY {
      /* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 25 */ } )

/* xsd:anyUri */

AnyURI ::= XMLStringWithNoCRLFHT
      (CONSTRAINED BY {
          /* La XMLStringWithNoCRLFHT doit être une URI valide comme défini
             dans IETF RFC 2396 */ } )

/* xsd:base64Binary */

Base64Binary ::= OCTET STRING

/* xsd:boolean */

Boolean ::= BOOLEAN

/* xsd:byte */

Byte ::= INTEGER (-128..127)

/* xsd:date */

Date ::= DATE-TIME (DateOnly)

/* xsd:dateTime */

DateTime ::= DATE-TIME

/* xsd:decimal */

Decimal ::= REAL (WITH COMPONENTS {..., base(10)})
                 (ALL EXCEPT(-0 | MINUS-INFINITY | PLUS-INFINITY | NOT-A-NUMBER))
```

```
/* xsd:double */

Double ::=  REAL (WITH COMPONENTS {
                      mantissa(-9007199254740991..9007199254740991),
                      base(2),
                      exponent(-1075..970)})

/* xsd:duration */

Duration ::= DURATION

/* xsd:ENTITIES */

ENTITIES ::= SEQUENCE (SIZE(1..MAX)) OF ENTITY

/* xsd:ENIITY */

ENTITY ::= NCName

/* xsd:float */

Float ::=  REAL (WITH COMPONENTS {
                      mantissa(-16777215..16777215),
                      base(2),
                      exponent(-149..104)})

/* xsd:gDay */

GDay ::= DATE-TIME (Day)

/* xsd:gMonth */

GMonth ::= DATE-TIME (Month)

/* xsd:gMonthDay */

GMonthDay ::= DATE-TIME (MonthDay)

/* xsd:gYear */

GYear ::= DATE-TIME (Year)

/* xsd:gYearMonth */

GYearMonth ::= DATE-TIME (YearMonth)

/* xsd:hexBinary */

HexBinary ::= OCTET STRING

/* xsd:ID */

ID ::= NCName

/* xsd:IDREF */

IDREF ::= NCName

/* xsd:IDREFS */

IDREFS ::= SEQUENCE (SIZE(1..MAX)) OF IDREF

/* xsd:int */

Int ::= INTEGER (-2147483648..2147483647)

/* xsd:integer */

Integer ::= INTEGER

/* xsd:language */

Language ::= VisibleString (FROM ("a".."z" | "A".."Z" | "-" | "0".."9"))
      (PATTERN
       "[a-zA-Z]#(1,8)(-[a-zA-Z0-9]#(1,8))*")

       /* La sémantique du langage est spécifiée dans IETF RFC 3066 */

/* xsd:long */

Long ::= INTEGER (-9223372036854775808..9223372036854775807)

/* xsd:name */
```

```
Name ::= Token (XMLStringWithNoWhitespace)
      (CONSTRAINED BY {
          /* Le Token doit être un Nom comme défini dans W3C XML 1.0, 2.3 */ } )

/* xsd:NCName */

NCName ::= Name
      (CONSTRAINED BY {
          /* Le Nom doit être un NCName comme défini dans W3C XML Namespaces,
             2 */ } )

/* xsd:negativeInteger */

NegativeInteger ::= INTEGER (MIN..-1)

/* xsd:NMTOKEN */

NMTOKEN ::= Token (XMLStringWithNoWhitespace)
      (CONSTRAINED BY {
           /* Le Token doit être un NMTOKEN comme défini dans W3C XML 1.0, 2.3 */ }
              )

/* xsd:NMTOKENS */

NMTOKENS ::= SEQUENCE (SIZE(1..MAX)) OF NMTOKEN

/* xsd:nonNegativeInteger */

NonNegativeInteger ::= INTEGER (0..MAX)

/* xsd:nonPositiveInteger */

NonPositiveInteger ::= INTEGER (MIN..0)

/* xsd:normalizedString */

NormalizedString ::= String (XMLStringWithNoCRLFHT)
      (CONSTRAINED BY {
          /* La Chaîne doit être une normalizedString comme défini dans
             le schéma XML du W3C Partie 2, 3.3.1 */})

/* xsd:NOTATION */

NOTATION ::= QName

/* xsd:positiveInteger */

PositiveInteger ::= INTEGER (1..MAX)

/* xsd:QName */

QName ::= SEQUENCE {
      uri   AnyURI OPTIONAL,
      name  NCName }

/* xsd:short */

Short ::= INTEGER (-32768..32767)

/* xsd:string */

String ::=  XMLCompatibleString

/* xsd:time */

Time ::= DATE-TIME (TimeOnly)

/* xsd:token */

Token ::= NormalizedString (CONSTRAINED BY {
      /* La NormalizedString doit être un token comme défini dans le XML du schéma
         W3C Partie 2, 3.3.2 */})

/* xsd:unsignedByte */

UnsignedByte ::=  INTEGER (0..255)

/* xsd:unsignedInt */

UnsignedInt ::=  INTEGER (0..4294967295)

/* xsd:unsignedLong */
```

```
UnsignedLong ::=  INTEGER (0..18446744073709551615)

/* xsd:unsignedShort */

UnsignedShort ::= INTEGER (0..65535)


/* Définitions de type ASN.1 servant de support au mappage des datatypes incorporés
du schéma XML du W3C */

XMLCompatibleString ::= UTF8String (FROM(
      {0, 0, 0, 9} |
      {0, 0, 0, 10} |
      {0, 0, 0, 13} |
      {0, 0, 0, 32} .. {0, 0, 215, 255} |
      {0, 0, 224, 0} .. {0, 0, 255, 253} |
      {0, 1, 0, 0} .. {0, 16, 255, 253}))

XMLStringWithNoWhitespace ::= UTF8String (FROM(
      {0, 0, 0, 33} .. {0, 0, 215, 255} |
      {0, 0, 224, 0} .. {0, 0, 255, 253} |
      {0, 1, 0, 0} .. {0, 16, 255, 253}))

XMLStringWithNoCRLFHT ::= UTF8String (FROM(
      {0, 0, 0, 32} .. {0, 0, 215, 255} |
      {0, 0, 224, 0} .. {0, 0, 255, 253} |
      {0, 1, 0, 0} .. {0, 16, 255, 253}))


/* Définitions de type ASN.1 servant de support au mappage des datatypes incorporés
de date et heure du schéma XML du W3C */

DURATION ::= VisibleString   (DE ("0".."9" | "DHMPSTY:.+-"))
                        (CONTRAINT PAR {/* Schéma XML du W3C Partie 2, 3.2.6 */})

DATE-TIME ::= VisibleString  (DE ("0".."9" | "TZ:.+-"))
                        (CONTRAINT PAR {/* Schéma XML du W3C Partie 2, 3.2.7 */})

DateOnly ::= DATE-TIME (DE ("0".."9" | "Z:+-"))
                        (CONTRAINT PAR {/* Schéma XML du W3C Partie 2, 3.2.9 */})

Day ::= DATE-TIME       (DE ("0".."9" | "Z:+-"))
                        (CONTRAINT PAR {/* Schéma XML du W3C Partie 2, 3.2.13 */})

Month ::= DATE-TIME     (DE ("0".."9" | "Z:+-"))

                        (CONTRAINT PAR {/* Schéma XML du W3C Partie 2, 3.2.14 */})

MonthDay ::= DATE-TIME (DE ("0".."9" | "Z:+-"))
                        (CONTRAINT PAR {/* Schéma XML du W3C Partie 2, 3.2.12 */})

Year ::= DATE-TIME      (DE ("0".."9" | "Z:+-"))
                        (CONTRAINT PAR {/* Schéma XML du W3C Partie 2, 3.2.11 */})

YearMonth ::= DATE-TIME (DE ("0".."9" | "Z:+-"))
                        (CONTRAINT PAR {/* Schéma XML du W3C Partie 2, 3.2.10 */})

TimeOnly ::= DATE-TIME (DE ("0".."9" | "Z:.+-"))
                        (CONTRAINT PAR {/* Schéma XML du W3C Partie 2, 3.2.8 */})

ENCODING-CONTROL XER
            GLOBAL-DEFAULTS MODIFIED-ENCODINGS
            GLOBAL-DEFAULTS CONTROL-NAMESPACE
                  "http://www.w3.org/2001/XMLSchema-instance"
                  PREFIX "xsi"
            NAMESPACE ALL, ALL IN ALL AS
                  "http://www.w3.org/2001/XMLSchema"
                  PREFIX "xsd"
            USE-QNAME QName
            BASE64 Base64Binary
            DECIMAL Decimal
            LIST ENTITIES, IDREFS, NMTOKENS
            EMBED-VALUES AnyType
            ANY-ATTRIBUTES AnyType.any-attributes
            ANY-ELEMENT AnyType.any-elements.*
            UNTAGGED AnyType.any-elements
```

```
                NAME AnySimpleType, AnyURI, Base64Binary, Boolean,
                     Byte, Date, DateTime, Decimal, Double, Duration,
                     Float, GDay, GMonth, GMonthDay, GYear, GYearMonth,
                     HexBinary, Int, Integer, Language, Long,
                     NegativeInteger, NonNegativeInteger, NonPositiveInteger,
                     NormalizedString, PositiveInteger, Short,
                     String, Time, Token,
                     UnsignedByte, UnsignedInt, UnsignedLong, UnsignedShort
                     AS UNCAPITALIZED
                WHITESPACE AnyURI, Language, Token, DURATION, DATE-TIME COLLAPSE
                WHITESPACE NormalizedString REPLACE

        END
```

Annexe B

Allocation des valeurs d'identifiant d'objet

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

La valeur suivante d'identifiant d'objet et de descripteur d'objet est attribuée dans la présente Recommandation | Norme internationale:

Pour le module définissant les types ASN.1 correspondant aux datatypes incorporés en XSD:

```
{ joint-iso-itu-t asn1(1) specification(0) modules(0) xsd-module(2) }
"ASN.1 XSD Module"
```

Annexe C

Exemples de mappages

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

La présente annexe illustre le mappage spécifié dans la présente Recommandation | Norme internationale en donnant un module ASN.1 correspondant à un schéma XSD.

C.1 Schéma utilisant des définitions de type simple

Le schéma suivant contient des exemples de datatypes incorporés en XSD (**xsd:string**, **xsd:decimal**, **xsd:integer**, **xsd:int**, **xsd:date**), d'autres **définition**s **de type simple** et de **définition**s **de type complexe**.

```
<?xml version="1.0" encoding="UTF-8"?>
	<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified">
		<xsd:element name="EXAMPLES">
			<xsd:complexType>
				<xsd:sequence>
					<xsd:element ref="personnelRecord"/>
					<xsd:element name="decimal" type="xsd:decimal"/>
					<xsd:element name="daysOfTheWeek" type="ListOfDays"/>
					<xsd:element ref="namesOfMemberNations"/>
					<xsd:element ref="fileIdentifier" maxOccurs="unbounded"/>
				</xsd:sequence>
			</xsd:complexType>
		</xsd:element>
		<xsd:element name="personnelRecord">
			<xsd:complexType>
				<xsd:sequence>
					<xsd:element name="name" type="name"/>
					<xsd:element name="title" type="xsd:string"/>
					<xsd:element name="decimal" type="xsd:integer"/>
					<xsd:element name="dateOfHire" type="xsd:date"/>
					<xsd:element ref="nameOfSpouse"/>
					<xsd:element ref="children"/>
				</xsd:sequence>
			</xsd:complexType>
		</xsd:element>
		<xsd:element name="nameOfSpouse" type="name"/>
		<xsd:complexType name="name">
			<xsd:sequence>
				<xsd:element name="givenName" type="xsd:string"/>
				<xsd:element name="initial" type="xsd:string"/>
				<xsd:element name="familyName" type="xsd:string"/>
			</xsd:sequence>
		</xsd:complexType>
		<xsd:element name="children">
			<xsd:complexType>
				<xsd:sequence>
					<xsd:element ref="ChildInformation" minOccurs="0"
maxOccurs="unbounded"/>
				</xsd:sequence>
			</xsd:complexType>
		</xsd:element>
		<xsd:element name="ChildInformation">
			<xsd:complexType>
				<xsd:sequence>
					<xsd:element name="name" type="name"/>
					<xsd:element name="dateOfBirth" type="xsd:date"/>
				</xsd:sequence>
			</xsd:complexType>
		</xsd:element>
		<xsd:simpleType name="ListOfDays">
			<xsd:list itemType="Day"/>
		</xsd:simpleType>
		<xsd:simpleType name="Day">
			<xsd:restriction base="xsd:string">
				<xsd:enumeration value="monday"/>
				<xsd:enumeration value="tuesday"/>
				<xsd:enumeration value="wednesday"/>
				<xsd:enumeration value="thursday"/>
				<xsd:enumeration value="friday"/>
				<xsd:enumeration value="saturday"/>
```

```
                        <xsd:enumeration value="sunday"/>
                    </xsd:restriction>
                </xsd:simpleType>
                <xsd:element name="namesOfMemberNations">
                    <xsd:simpleType>
                        <xsd:list itemType="xsd:string"/>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="fileIdentifier">
                    <xsd:complexType>
                        <xsd:choice>
                            <xsd:element name="serialNumber" type="xsd:int"/>
                            <xsd:element name="relativeName" type="xsd:string"/>
                            <xsd:element ref="unidentified"/>
                        </xsd:choice>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name="unidentified">
                    <xsd:complexType>
                        <xsd:complexContent>
                            <xsd:restriction base="xsd:anyType"/>
                        </xsd:complexContent>
                    </xsd:complexType>
                </xsd:element>
            </xsd:schema>
```

C.2 Définitions ASN.1 correspondantes

Ce qui suit est la spécification correspondante en ASN.1 et valide les mêmes documents XML que le schéma XSD:

```
EXAMPLES{joint-iso-itu-t asn1(1) examples(999) xml-defined-types(3)}
DEFINITIONS AUTOMATIC TAGS

XER INSTRUCTIONS ::=
BEGIN

IMPORTS String, Decimal, Int, Date, AnyType

      FROM XSD
        {joint-iso-itu-t asn1(1) specification(0) modules(0) xsd-module(1)};

EXAMPLES ::=      SEQUENCE {
      personnelRecord      PersonnelRecord,
      number               Decimal,
      daysOfTheWeek        ListOfDays,
      namesOfMemberNations NamesOfMemberNations,
      fileIdentifier-list  [UNTAGGED]
             SEQUENCE (SIZE(1..MAX)) OF fileidentifier FileIdentifier }

PersonnelRecord  ::=   [NAME AS UNCAPITALIZED] SEQUENCE {
      name             Name,
      title            XSD.String,
      number           INTEGER,
      dateOfHire       Date,
      nameOfSpouse     NameOfSpouse,
      children         Children }

NameOfSpouse ::= [NAME AS UNCAPITALIZED] Name

Name ::= [NAME AS UNCAPITALIZED] SEQUENCE {
      givenName        XSD.String,
      initial          XSD.String,
      familyName       XSD.String }

Children ::= [NAME AS UNCAPITALIZED] SEQUENCE {
       childInformation-list [UNTAGGED]
            SEQUENCE OF ChildInformation }

ChildInformation ::=   SEQUENCE {
      name             Name,
      dateOfBirth      Date }
```

```
ListOfDays ::=    [LIST] SEQUENCE OF Day

Day ::=     ENUMERATED {monday, tuesday, wednesday, thursday, friday,
                  saturday, sunday}

NamesOfMemberNations ::= [NAME AS UNCAPITALIZED] [LIST] SEQUENCE OF XSD.String

FileIdentifier ::= [NAME AS UNCAPITALIZED] SEQUENCE {
      choice [UNTAGGED] CHOICE {
            serialNumber     Int,
            relativeName     XSD.String,
            unidentified     UNIDENTIFIED   } }

UNIDENTIFIED ::=  [NAME AS LOWERCASED] XSD.AnyType

ENCODING-CONTROL XER
      GLOBAL-DEFAULTS MODIFIED-ENCODINGS

END
```

C.3 Autres exemples

Dans ce paragraphe, tous les exemples partiels (les exemples qui ne contiennent pas l'élément **schéma**) supposent que les éléments XML représentant la syntaxe XSD sont dans le domaine d'application d'une déclaration de **namespace par défaut** dont le **nom de namespace** est le **namespace cible** du schéma.

C.3.1 Documents schéma avec les items d'information d'élément import et include

Le schéma XSD suivant se compose de deux namespaces qui sont composés à partir de quatre fichiers schéma:

```
<!-- file "http://example.com/xyz/schema.xsd" -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xyz="http://example.com/xyz"
        targetNamespace="http://example.com/xyz">

        <xsd:element name="xyz-elem" type="xsd:string"/>

        <xsd:complexType name="Xyz-type">
                <xsd:attribute name="xyz-attr" type="xsd:boolean"/>
        </xsd:complexType>
</xsd:schema>

<!-- file "http://example.com/abc/main.xsd" -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:abc="http://example.com/abc"
        targetNamespace="http://example.com/xyz">

        <xsd:include schemaLocation="http://example.com/abc/sub1.xsd"/>

        <xsd:import namespace="http://www.w3.org/2001/XMLSchema"
                schemaLocation="http://example.com/xyz/schema.xsd"/>

        <xsd:redefine schemaLocation="http://example.com/abc/sub2.xsd">
                <xsd:attribute name="sub2-attr" type="xsd:token"/>
        </xsd:redefine>

        <xsd:element name="abc-elem" type="Xyz-type"/>
</xsd:schema>

<!-- file "http://example.com/abc/sub1.xsd" -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:abc="http://example.com/abc"
              targetNamespace="http://example.com/xyz">

        <xsd:element name="sub1-elem" type="xsd:string"/>
</xsd:schema>

<!-- file "http://example.com/abc/sub2.xsd" -->

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <xsd:element name="sub2-elem " type="xsd:string"/>
        <xsd:attribute name="sub2-attr" type="xsd:string"/>
</xsd:schema>
```

Ces quatre documents schéma sont mappés dans les deux modules ASN.1 suivants:

```
XYZ   -- La référence du module n'est pas normalisée
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN

Xyz-elem ::= [NAME AS UNCAPITALIZED] XSD.String
Xyz-type ::= SEQUENCE {
      xyz-attr [ATTRIBUTE] BOOLEAN OPTIONAL }

ENCODING-CONTROL XER
      GLOBAL-DEFAULTS MODIFIED-ENCODINGS
      GLOBAL-DEFAULTS CONTROL-NAMESPACE
            "http://www.w3.org/2001/XMLSchema-instance"
END

ABC   -- La référence du module n'est pas normalisée
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
IMPORTS
Xyz-type FROM XYZ
Token, String FROM XSD;

Abc-elem ::= [NAME AS UNCAPITALIZED] Xyz-type
Sub1-elem ::= [NAME AS UNCAPITALIZED] XSD.String
Sub2-elem ::= [NAME AS UNCAPITALIZED] XSD.String
Sub2-attr ::= [NAME AS UNCAPITALIZED] [ATTRIBUTE] XSD.Token

ENCODING-CONTROL XER
      GLOBAL-DEFAULTS MODIFIED-ENCODINGS
      GLOBAL-DEFAULTS CONTROL-NAMESPACE
            "http://www.w3.org/2001/XMLSchema-instance"
END
```

C.3.2 Mappage des définitions de type simple

C.3.2.1 définition de type simple déduite par restriction

Pour un ensemble complet d'exemples de restrictions de type simple, voir les exemples de facettes au § C.3.3.

C.3.2.2 définition de type simple déduite par liste

```
<xsd:simpleType name="Int-list">
      <xsd:list itemType="xsd:integer"/>
</xsd:simpleType>

<xsd:simpleType name="Int-10-to-100-list">
      <xsd:list>
            <xsd:simpleType>
                  <xsd:restriction base="xsd:integer">
                        <xsd:minInclusive value="10"/>
                        <xsd:minInclusive value="100"/>
                  </xsd:restriction>
            </xsd:simpleType>
      </xsd:list
</xsd:simpleType>
```

Ces définitions de type simple sont mappées dans les allocations de type ASN.1 suivantes:

```
Int-list ::= [LIST] SEQUENCE OF INTEGER

Int-10-to-100-list ::= [LIST] SEQUENCE OF INTEGER (10..100)
```

C.3.2.3 définition de type simple déduite par union

```
<xsd:simpleType name="Int-or-boolean">
      <xsd:union itemType="xsd:integer xsd:boolean"/>
</xsd:simpleType>

<xsd:simpleType name="Time-or-int-or-boolean--or-dateRestriction">
      <xsd:union itemType=" xsd:time Int-or-boolean">
            <xsd:simpleType>
                  <xsd:restriction base="xsd:date">
                        <xsd:minInclusive value="2003-01-01"/>
```

```
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:union>
        </xsd:simpleType>
```

Ces **définitions de type simple** sont mappées dans les allocations de type ASN.1 suivantes:

```
Int-or-boolean ::= [USE-UNION] CHOICE {
      integer [NAMESPACE "http://www.w3.org/2001/XMLSchema"] INTEGER,
      boolean [NAMESPACE "http://www.w3.org/2001/XMLSchema"] BOOLEAN }

Time-or-int-or-boolean-or-dateRestriction ::= [USE-UNION] CHOICE {
      time [NAMESPACE "http://www.w3.org/2001/XMLSchema"] XSD.Time,
      integer [NAMESPACE "http://www.w3.org/2001/XMLSchema"] INTEGER,
      boolean [NAMESPACE "http://www.w3.org/2001/XMLSchema"] BOOLEAN,
      alt [NAME AS ""] XSD.Date (CONSTRAINED BY
            { /* minInclusive="2003-01-01" */ }) }
```

C.3.2.4 Mappage des hiérarchies de déduction de type pour les définitions de type simple

```
<xsd:simpleType name="Int-10-to-50">
        <xsd:restriction base="xsd:integer">
                <xsd:minExclusive value="10"/>
                <xsd:maxExclusive value="50"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Ten-multiples">
        <xsd:restriction base="Int-10-to-50">
                <xsd:enumeration value="20"/>
                <xsd:enumeration value="30"/>
                <xsd:enumeration value="40"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Twenty-multiples">
        <xsd:restriction base="Ten-multiples">
                <xsd:pattern value=".*[02468]0|0"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="Stock-level">
        <xsd:extension base="Int-10-to-50">
                <xsd:attribute name="procurement" type="Int-10-to-50"/>
        </xsd:extension>
</xsd:complexType>
```

Ces **définitions de type simple** sont mappées dans les allocations de type ASN.1 suivantes:

```
Int-10-to-50 ::= INTEGER (10<..<50)

Ten-multiples ::= ENUMERATED {int20(20), int30(30), int40(40)}

Twenty-multiples ::= ENUMERATED {int20(20), int40(40)}

Stock-level ::= SEQUENCE {
      procurement [ATTRIBUTE] Int-10-to-50 OPTIONAL,
      base        Int-10-to-50-derivations }

Ten-multiples-derivations ::= [USE-TYPE] CHOICE {
      ten-multiples    [NAME AS CAPITALIZED] Ten-multiples,
      twenty-multiples [NAME AS CAPITALIZED] Twenty-multiples }

Int-10-to-50-derivations ::= [USE-TYPE] CHOICE {
      int-10-to-50     [NAME AS CAPITALIZED] Int-10-to-50,
      ten-multiples    [NAME AS CAPITALIZED] Ten-multiples,
      twenty-multiples [NAME AS CAPITALIZED] Twenty-multiples,
      stock-level      [NAME AS CAPITALIZED] Stock-level }
```

C.3.3 Mappage des facettes

C.3.3.1 length, minLength et maxLength

```
<xsd:simpleType name="String-10">
        <xsd:restriction base="xsd:string">
                <xsd:length value="10"/>
        </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="String-5-to-10">
        <xsd:restriction base="xsd:string">
                <xsd:minLength value="5"/>
                <xsd:maxLength value="10"/>
        </xsd:restriction>
</xsd:simpleType>
```

Ces deux **définitions de type simple** sont mappées dans les allocations de type ASN.1 suivantes:

```
String-10 ::= XSD.String (SIZE(10))

String-5-to-10 ::= XSD.String (SIZE(5..10))
```

C.3.3.2 pattern

```
<xsd:simpleType name="My-filename">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="[&#00;-&#FF;]*"/>
                <xsd:pattern value="/?([^/]*/)*[^/]*/*" />
        </xsd:restriction>
</xsd:simpleType>
```

Cette **définition de type simple** est mappée dans l'allocation de type ASN.1 suivante:

```
My-filename ::= XSD.String
      (CONSTRAINED BY
            {/* représentation XML du pattern XSD
                "&#00;-&#FF;" | "/?([^/]*&#x2F;)*[^/]*&#x2F;*" */})
```

C.3.3.3 whiteSpace

```
<xsd:simpleType name="My-String">
        <xsd:restriction base="xsd:string">
                <xsd:whitespace value="preserve"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="My-NormalizedString">
        <xsd:restriction base="xsd:string">
                <xsd:whitespace value="replace"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="My-TokenString">
        <xsd:restriction base="xsd:string">
                <xsd:whitespace value="collapse"/>
        </xsd:restriction>
</xsd:simpleType>
```

Ces **définitions de type simple** sont mappées dans les allocations de type ASN.1 suivantes:

```
My-String ::= XSD.String

My-NormalizedString ::= [WHITESPACE REPLACE] XSD.String
            (FROM ({0, 0, 0, 32} .. {0, 16, 255, 255}))

My-TokenString ::= [WHITESPACE REPLACE] XSD.String
            (FROM ({0, 0, 0, 32} .. {0, 16, 255, 255}))
            (PATTERN "([^ ]([^ ]| [^ ])*)?")
```

C.3.3.4 minInclusive, minExclusive, maxInclusive et maxExclusive

```
<xsd:simpleType name="Int-10-to-100">
        <xsd:restriction base="xsd:integer">
                <xsd:minExclusive value="10"/>
                <xsd:maxInclusive value="100"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Pi-approximation">
        <xsd:restriction base="xsd:double">
                <xsd:minExclusive value="3.14159"/>
                <xsd:maxExclusive value="3.1416"/>
        </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:simpleType name="Morning">
        <xsd:restriction base="xsd:time">
                <xsd:minInclusive value="00:00:00"/>
                <xsd:maxExclusive value="12:00:00"/>
        </xsd:restriction>
</xsd:simpleType>
```

Ces **définitions de type simple** sont mappées dans les allocations de type ASN.1 suivantes:

```
Int-10-to-100 ::= INTEGER (10<..100)

Pi-approximation ::= XSD.Double (3.14159<..<3.1416)

Morning ::= XSD.Time (CONSTRAINED BY
      {/* minInclusive="00:00:00" maxExclusive="12:00:00" */ })
```

C.3.3.5 totalDigits et fractionDigits

```
<xsd:simpleType name="RefundableExpenses">
        <xsd:restriction base="xsd:decimal">
                <xsd:totalDigits="5"/>
                <xsd:fractionDigits value="2"/>
        </xsd:restriction>
</xsd:simpleType>
```

Cette **définition de type simple** est mappée dans l'allocation de type ASN.1 suivante:

```
RefundableExpenses ::= XSD.Decimal (CONSTRAINED BY
      {/* totalDigits="5" fractionDigits="2" */ })
```

C.3.3.6 enumeration

```
<xsd:simpleType name="FarmAnimals">
        <xsd:restriction base="xsd:normalizedString">
                <xsd:enumeration value="Horse"/>
                <xsd:enumeration value="Bull"/>
                <xsd:enumeration value="Cow"/>
                <xsd:enumeration value="Pig"/>
                <xsd:enumeration value="Duck"/>
                <xsd:enumeration value="Goose"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="PrimeNumbersBelow30">
        <xsd:restriction base="xsd:integer">
                <xsd:enumeration value="2"/>
                <xsd:enumeration value="3"/>
                <xsd:enumeration value="5"/>
                <xsd:enumeration value="7"/>
                <xsd:enumeration value="11"/>
                <xsd:enumeration value="13"/>
                <xsd:enumeration value="17"/>
                <xsd:enumeration value="19"/>
                <xsd:enumeration value="23"/>
                <xsd:enumeration value="29"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="X680-release">
        <xsd:restriction base="xsd:gYearMonth">
                <xsd:enumeration value="2002-07"/>
                <xsd:enumeration value="1997-12"/>
                <xsd:enumeration value="1994-07"/>
        </xsd:restriction>
</xsd:simpleType>
```

Ces **définitions de type simple** sont mappées dans les allocations de type ASN.1 suivantes:

```
FarmAnimals ::= ENUMERATED {horse, bull, cow, pig, duck, goose}

PrimeNumbersBelow30 ::= [USE-NUMBER] ENUMERATED {int2(2), int3(3), int5(5),
      int7(7), int11(11), int13(13), int17(17), int19(19), int23(23), int29(29)}

X680-release ::= XSD.GYearMonth ("2002-07" | "1997-12" | "1994-07")
```

L'instruction de codage suivante est incluse dans la section de contrôle de codage XER:

```
TEXT FarmAnimals:ALL AS CAPITALIZED
```

C.3.3.7 enumeration en conjonction avec d'autres facettes

Les exemples suivants sont fondés sur l'héritage des facettes utilisant la restriction de certains des types définis au § C.3.3.6.

```
<xsd:simpleType name="FarmAnimals-subset">
        <xsd:restriction base="FarmAnimals">
                <xsd:minLength value="4"/>
                <xsd:pattern value="[^oe]*"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="PrimeNumbersBelow30-subset">
        <xsd:restriction base="PrimeNumbersBelow30">
                <xsd:minExclusive value="5"/>
                <xsd:pattern value=".*[23].*"/>
        </xsd:restriction>
</xsd:simpleType>
```

Ces **définitions de type simple** sont mappées dans les allocations de type ASN.1 suivantes:

```
/* Cheval et Oie ne satisfont pas à la facette pattern

   Vache et Cochon ne satisfont pas à la facette minLength */

FarmAnimals-subset ::= ENUMERATED {bull, duck}

/* 2, 3 et 5 ne satisfont pas à la facette minExclusive

   2, 5, 7, 11, 17 et 19 ne satisfont pas à la facette pattern */

PrimeNumbersBelow30-subset ::= [USE-NUMBER] ENUMERATED {int13(13), int23(23),
                                                        int29(29)}
```

L'instruction de codage suivante est incluse dans la section de contrôle de codage XER:

```
TEXT FarmAnimals-subset:ALL AS CAPITALIZED
```

C.3.4 Mappage des déclarations d'élément

C.3.4.1 déclarations d'élément dont la définition de type est une définition de type simple ou une définition de type complexe de niveau supérieur définie par l'usager

```
<xsd:element name="Forename" type="xsd:token"/>

<xsd:element name="File" type="My-filename"/>

<xsd:element name="Value" type="Int-10-to-50"/>
```

Ces **déclarations d'élément** sont mappées dans les allocations de type ASN.1 suivantes:

```
Forename ::= XSD.Token

File ::= My-filename

Value ::= Int-10-to-50-derivations
```

NOTE – Le type "**My-filename**" et son mappage en ASN.1 sont définis au § C.3.3.2; le type "**Int-10-to-50**" et son mappage en ASN.1 sont définis au § C.3.2.4.

C.3.4.2 déclarations d'élément dont la définition de type est une définition de type simple ou définition de type complexe anonyme

```
<xsd:element name="maxOccurs">
        <xsd:simpleType>
                <xsd:union memberTypes="xsd:nonNegativeInteger">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:token">
                                        <xsd:enumeration name="unbounded"/>
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:union>
        </xsd:simpleType>
</xsd:element>

<xsd:element name="address">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="line-1" type="xsd:token"/>
                        <xsd:element name="line-2" type="xsd:token"/>
                        <xsd:element name="city" type="xsd:token"/>
```

```
                <xsd:element name="state" type="xsd:token" minOccurs="0"/>
                <xsd:element name="zip" type="xsd:token"/>
            </xsd:sequence>
            <xsd:attribute name="country" type="xsd:token"/>
        </xsd:complexType>
    </xsd:element>
```

Ces **déclarations d'élément** sont mappées dans les allocations de type ASN.1 suivantes:

```
MaxOccurs ::= [NAME AS UNCAPITALIZED] [USE-UNION] CHOICE {
      nonNegativeInteger [NAMESPACE AS "http://www.w3.org/2001/XMLSchema"]
                                        XSD.NonNegativeInteger,
      alt                [NAME AS ""]    ENUMERATED {unbounded} }

Address ::= [NAME AS UNCAPITALIZED] SEQUENCE {
      Country      [ATTRIBUTE] XSD.Token OPTIONAL,
      line-1       XSD.Token,
      line-2       XSD.Token,
      city         XSD.Token,
      state        XSD.Token OPTIONAL,
      zip          XSD.Token }
```

C.3.4.3 déclarations d'élément qui sont l'en-tête d'un groupe de substitution d'élément

```
<xsd:element name="Tic" type="xsd:integer"  abstract="true"/>

<xsd:element name="Tac" type="xsd:byte" substitutionGroup="Tic"/>

<xsd:element name="Toe" substitutionGroup="Tic"/>

<xsd:element name="Foo" type="xsd:date"/>

<xsd:element name="Bar" substitutionGroup="Foo"/>
```

Ces **déclarations d'élément** sont mappées en:

```
Tac ::=  INTEGER (-128..127)

Toe ::= INTEGER

Tic-group ::= [UNTAGGED] CHOICE {
      tac [NAME AS CAPITALIZED] Tac,
      toe [NAME AS CAPITALIZED] Toe }

Foo ::= XSD.Date

Bar ::= XSD.Date

Foo-group ::= [UNTAGGED] CHOICE {
      foo [NAME AS CAPITALIZED] Foo,
      bar [NAME AS CAPITALIZED] Bar }
```

C.3.4.4 déclarations d'élément avec une contrainte de valeur qui est une valeur par défaut

C.3.4.4.1 Ce qui suit est une **déclaration d'élément** avec une **définition de type simple** anonyme, non utilisée comme **définition de type de base** de quelque type que ce soit.

```
<xsd:element name="Telephone" type="xsd:token" default="undefined"/>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
Telephone ::= [DEFAULT-FOR-EMPTY "undefined"] XSD.Token
```

C.3.4.4.2 Ce qui suit est une **déclaration d'élément** avec une **définition de type complexe** anonyme avec contenu simple, non utilisée comme **définition de type de base** de quelque type que ce soit.

```
<xsd:element name="InternationalTelephone" default="undefined">
    <xsd:simpleContent>
        <xsd:extension base="xsd:token">
            <xsd:attribute name="country-code" type="xsd:integer"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:element>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
InternationalTelephone ::= [DEFAULT-FOR-EMPTY "undefined"] SEQUENCE {
      country-code  [ATTRIBUTE] INTEGER OPTIONAL,
      base          [UNTAGGED]  XSD.Token }
```

C.3.4.4.3 Ce qui suit est une **déclaration d'élément** avec une **définition de type complexe** anonyme. La **définition de type complexe** a un contenu complexe qui est **mixte** et vidable et n'est pas utilisée comme **définition de type de base** de quelque type que ce soit.

```
<xsd:element name="Description" default="absent" mixed="true">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="bold" type="xsd:string"/>
            <xsd:element name="italic" type="xsd:string"/>
      </xsd:choice>
</xsd:element>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
Description ::= [EMBED-VALUES] [DEFAULT-FOR-EMPTY "absent"] SEQUENCE {
      embed-values  SEQUENCE OF XSD.String,
      choice-list   [UNTAGGED] SEQUENCE OF [UNTAGGED] CHOICE {
            bold        XSD.String,
            italic      XSD.String } } (CONSTRAINED BY
      {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 25 */})
```

C.3.4.4.4 La **définition de type** de la **déclaration d'élément** dans l'exemple suivant est utilisée comme **définition de type de base** d'un autre type.

Cet exemple utilise les types XSD et ASN.1 de l'exemple du § C.3.2.4.

```
<xsd:element name="Quantity" type="Int-10-to-50" default="20"/>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
Quantity ::= Int-10-to-50-deriv-default-20
```

Si aucun type ASN.1 correspondant à `Int-10-to-50`, avec une valeur par défaut de "20" n'a déjà été généré, le type suivant est aussi généré:

```
Int-10-to-50-deriv-default-20 ::= [USE-TYPE] CHOICE {
      int-10-to-50      [NAME AS CAPITALIZED] [DEFAULT-FOR-EMPTY 20]
                                    Int-10-to-50,
      ten-multiples     [NAME AS CAPITALIZED] [DEFAULT-FOR-EMPTY int20]
                                    Ten-multiples,
      twenty-multiples  [NAME AS CAPITALIZED] [DEFAULT-FOR-EMPTY int20]
                                    Twenty-multiples,
      stock-level       [NAME AS CAPITALIZED] [DEFAULT-FOR-EMPTY 20]
                                    Stock-level }
```

C.3.4.5 déclaration d'élément avec une contrainte de valeur qui est une valeur fixée

C.3.4.5.1 Ce qui suit est une **déclaration d'élément** avec une **définition de type simple** anonyme, non utilisée comme **définition de type de base** de quelque type que ce soit.

```
<xsd:element name="UnknownTelephone" type="xsd:token" fixed="undefined"/>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
UnknownTelephone ::= [DEFAULT-FOR-EMPTY "undefined"] XSD.Token ("undefined")
```

C.3.4.5.2 Ce qui suit est une **déclaration d'élément** avec une **définition de type complexe** anonyme. La **définition de type complexe** a un contenu simple et n'est pas utilisée comme **définition de type de base** de quelque type que ce soit.

```
<xsd:element name="UnknownInternationalTelephone" fixed="undefined">
      <xsd:simpleContent>
            <xsd:extension base="xsd:token">
                  <xsd:attribute name="country-code" type="xsd:integer"/>
            </xsd:extension>
      </xsd:simpleContent>
</xsd:element>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
UnknownInternationalTelephone ::= [DEFAULT-FOR-EMPTY "undefined"] SEQUENCE {
      country-code    [ATTRIBUTE] INTEGER OPTIONAL,
      base            [UNTAGGED] XSD.Token }
            (WITH COMPONENTS {..., base ("undefined")})
```

C.3.4.5.3 Ce qui suit est une **déclaration d'élément** avec une **définition de type complexe** anonyme. La **définition de type complexe** a un contenu complexe qui est **mixte** et vidable et n'est pas utilisée comme **définition de type de base** de quelque type que ce soit.

```
<xsd:element name="UnknownDescription" fixed="absent" mixed="true">
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="bold" type="xsd:string"/>
            <xsd:element name="italic" type="xsd:string"/>
      </xsd:choice>
</xsd:element>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
UnknownDescription ::= [EMBED-VALUES] [DEFAULT-FOR-EMPTY "absent"] SEQUENCE {
      embed-values      SEQUENCE OF XSD.String,
      choice-list       [UNTAGGED] SEQUENCE OF [UNTAGGED] CHOICE {
            bold              XSD.String,
            italic            XSD.String } }
      (CONSTRAINED BY
            {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 25 */})
      (WITH COMPONENTS {embed-values ({"absent"}),
                        choice-list (SIZE(0))})
```

C.3.4.5.4 La **définition de type** de la **déclaration d'élément** suivante est une **définition de type simple** utilisée comme **définition de type de base** d'un autre type.

Le présent exemple utilise les types XSD et ASN.1 de l'exemple du § C.3.2.4.

```
<xsd:element name="Quantity" type="Int-10-to-50" fixed="20"/>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
Quantity ::= Int-10-to-50-deriv-fixed-20
```

Si aucun type ASN.1 correspondant à `Int-10-to-50` avec une valeur fixée de "20" n'a déjà été généré, le type suivant est aussi généré:

```
Int-10-to-50-deriv-fixed-20 ::= [USE-TYPE] CHOICE {
      int-10-to-50      [NAME AS CAPITALIZED] [DEFAULT-FOR-EMPTY 20]
                              Int-10-to-50,
      ten-multiples     [NAME AS CAPITALIZED] [DEFAULT-FOR-EMPTY int20]
                              Ten-multiples,
      twenty-multiples  [NAME AS CAPITALIZED] [DEFAULT-FOR-EMPTY int20]
                              Twenty-multiples,
      stock-level       [NAME AS CAPITALIZED] [DEFAULT-FOR-EMPTY 20]
                              Stock-level }
(WITH COMPONENTS {
      int-10-to-50 (20),
      ten-multiples (int20),
      twenty-multiples (int20),
      stock-level (WITH COMPONENTS {..., base (20)}) })
```

C.3.4.6 déclarations d'élément qui sont nillables

C.3.4.6.1 L'exemple suivant montre une **déclaration d'élément** qui est **nillable** et dont la **définition de type** est un datatype incorporé en XSD.

```
<xsd:element name="Nillable-1" type="xsd:string" nillable="true"/>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
Nillable-1 ::= [USE-NIL] SEQUENCE {
      content XSD.String OPTIONAL }
```

C.3.4.6.2 L'exemple suivant montre une **déclaration d'élément** qui est **nillable** et dont la **définition de type** est une **définition de type complexe** anonyme.

```
<xsd:element name="Nillable-2" nillable="true">
      <xsd:complexType>
            <xsd:sequence>
                  <xsd:element name="a" type="xsd:string"/>
                  <xsd:element name="b" type="xsd:string"/>
            </xsd:sequence>
            <xsd:attribute name="b" type="xsd:boolean"/>
        </xsd:complexType>
</xsd:element>
```

Cette **déclaration d'élément** est mappée dans l'allocation de type ASN.1 suivante:

```
Nillable-2 ::= [USE-NIL] SEQUENCE {
      b           [ATTRIBUTE] BOOLEAN OPTIONAL,
      content     SEQUENCE {
            a           XSD.String,
            b           XSD.String } OPTIONAL }
```

C.3.4.6.3 L'exemple suivant montre une **déclaration d'élément** qui est **nillable** et dont la **définition de type** est une **définition de type complexe** de niveau supérieur.

```
<xsd:complexType name="Foo">
      <xsd:sequence>
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="b" type="xsd:boolean"/>
</xsd:complexType>

<xsd:element name="Nillable-3" type="Foo" nillable="true"/>
```

Ces composants de schéma sont mappés dans les allocations de type ASN.1 suivantes:

```
Foo ::= SEQUENCE {
      b     [ATTRIBUTE] BOOLEAN OPTIONAL,
      a     XSD.String,
      b-1   [NAME AS "b"] XSD.String }

Foo-nillable ::= [USE-NIL] SEQUENCE {
      b           [ATTRIBUTE] BOOLEAN OPTIONAL,
      content     SEQUENCE {
            a           XSD.String,
            b           XSD.String } OPTIONAL }

Nillable-3 ::= Foo-nillable
```

C.3.4.6.4 L'exemple suivant montre une **déclaration d'élément** qui est **nillable**, dont la **définition de type** est une **définition de type complexe** de niveau supérieur et qui est utilisée comme **définition de type de base** d'une autre **définition de type complexe**.

Les composants de schéma suivants sont définis et s'ajoutent aux composants de schéma du § C.3.4.6.3:

```
<xsd:complexType name="Bar">
      <xsd:complexContent>
            <xsd:extension base="Foo">
                  <xsd:sequence>
                        <xsd:element name="z" type="xsd:string"/>
                  </xsd:sequence>
                  <xsd:attribute name="c" type="xsd:boolean"/>
            </xsd:extension>
      </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Nillable-4" type="Foo" nillable="true"/>
```

Les types ASN.1 suivants sont générés en plus du type Foo du § C.3.4.6.3:

```
Bar ::= SEQUENCE {
      b     [ATTRIBUTE] BOOLEAN OPTIONAL,
      c     [ATTRIBUTE] BOOLEAN OPTIONAL,
      a     XSD.String,
      b-1   [NAME AS "b"] XSD.String,
      z     XSD.String }

Foo-derivations ::= [USE-TYPE] CHOICE {
      foo   [NAME AS CAPITALIZED] Foo,
      bar   [NAME AS CAPITALIZED] Bar }

Foo-nillable ::= [USE-NIL] SEQUENCE {
      b           [ATTRIBUTE] BOOLEAN OPTIONAL,
      content     SEQUENCE {
            a           XSD.String,
            b           XSD.String } OPTIONAL }
```

```
Bar-nillable ::= [USE-NIL] SEQUENCE {
      b           [ATTRIBUTE] BOOLEAN OPTIONAL,
      c           [ATTRIBUTE] BOOLEAN OPTIONAL,
      content     SEQUENCE {
            a           XSD.String,
            b           XSD.String,
            z           XSD.String } OPTIONAL }

Foo-deriv-nillable ::= [USE-TYPE] CHOICE {
      foo         [NAME AS CAPITALIZED] Foo-nillable,
      bar         [NAME AS CAPITALIZED] Bar-nillable }

Nillable-4 ::= Foo-deriv-nillable
```

C.3.5 Mappage des utilisations d'attributs et des déclarations d'attribut

C.3.5.1 Ce qui suit est un exemple de **déclaration d'attribut** de niveau supérieur dont la **définition de type** est une **définition de type simple** de niveau supérieur définie par l'usager.

```
<xsd:attribute name="name" type="NCName"/>
```

Cette **déclaration d'attribut** est mappée dans l'allocation de type ASN.1 suivante:

```
Name ::= [NAME AS UNCAPITALIZED] [ATTRIBUTE] XSD.NCName
```

C.3.5.2 Ce qui suit est un exemple de **déclaration d'attribut** de niveau supérieur dont la **définition de type** est une **définition de type simple** anonyme.

```
<xsd:attribute name="form">
      <xsd:simpleType>
            <xsd:restriction base="xsd:token">
                  <xsd:enumeration value="qualified"/>
                  <xsd:enumeration value="unqualified"/>
            </xsd:restriction>
      </xsd:simpleType>
</xsd:attribute>
```

Cette **déclaration d'attribut** est mappée dans l'allocation de type ASN.1 suivante:

```
Form ::= [NAME AS UNCAPITALIZED] [ATTRIBUTE] ENUMERATED {qualified, unqualified}
```

C.3.5.3 L'exemple suivant est une **utilisation d'attribut** avec une **contrainte de valeur** qui est une valeur par **défaut**.

La **déclaration d'attribut** dont le **nom** est "form" et qui est référencée dans cet exemple est définie au § C.3.5.2.

```
<xsd:complexType name="element">
      <xsd:attribute name="name" type="xsd:NCName" default="NAME"/>
      <xsd:attribute ref="form" default="qualified"/>
</xsd:complexType>
```

Cette **définition de type complexe** est mappée dans l'allocation de type ASN.1 suivante:

```
Element ::= [NAME AS UNCAPITALIZED] SEQUENCE {
      name [ATTRIBUTE] XSD.NCName DEFAULT "NAME",
      form [ATTRIBUTE] Form DEFAULT qualified }
```

C.3.5.4 Cet exemple montre une **déclaration d'attribut** de niveau supérieur avec une **contrainte de valeur** qui est une valeur par **défaut** et une **utilisation d'attribut** avec cette **déclaration d'attribut**.

```
<xsd:attribute name="minOccurs" type="xsd:nonNegativeInteger" default="1"/>

<xsd:attribute name="maxOccurs" default="1"/>
      <xsd:simpleType>
            <xsd:union memberTypes="xsd:nonNegativeInteger" >
                  <xsd:simpleType>
                        <xsd:restriction base="xsd:NMTOKEN">
                              <xsd:enumeration value="unbounded"/>
                        </xsd:restriction>
                  </xsd:simpleType>
            </xsd:union>
      </xsd:simpleType>
</xsd:attribute>
```

```
<xsd:complexType name="Particle">
      <xsd:sequence>
            <xsd:element name="particle"/>
      </xsd:sequence>
      <xsd:attribute ref="minOccurs"/>
      <xsd:attribute ref="maxOccurs" default="unbounded"/>
</xsd:complexType>
```

Ces composants de schéma sont mappés dans les allocations de type ASN.1 suivantes:

```
MinOccurs ::= [ATTRIBUTE] [NAME AS UNCAPITALIZED] XSD.NonNegativeInteger

MaxOccurs ::= [ATTRIBUTE] [NAME AS UNCAPITALIZED] [USE-UNION] CHOICE {
      nonNegativeInteger     [NAMESPACE AS "http://www.w3.org/2001/XMLSchema"]
                                    XSD.NonNegativeInteger,
      alt                    [NAME AS ""] ENUMERATED {unbounded} }

Particle ::= SEQUENCE {
      minOccurs         [ATTRIBUTE] MinOccurs DEFAULT 1,
      maxOccurs         [ATTRIBUTE] MaxOccurs DEFAULT alt : unbounded,
      particle         XSD.AnyType }
```

C.3.5.5 Cet exemple montre une **déclaration d'attribut** dont la **déclaration d'attribut** a un **namespace cible** qui n'est pas **absent**.

```
<xsd:complexType name="Ack">
      <xsd:attribute name="number" type="xsd:integer" form="qualified" />
</xsd:complexType>
```

Cette **définition de type complexe** est mappée dans l'allocation de type ASN.1 suivante:

```
Ack ::= SEQUENCE {
      number [NAMESPACE AS "http://targetnamespaceForExample"] [ATTRIBUTE]
                        INTEGER OPTIONAL }
```

C.3.6 Mappage des définitions de groupe modèle

C.3.6.1 Ce qui suit est une **définition de groupe modèle** dont le **groupe modèle** a un **composeur** de **sequence.**

```
<xsd:group name="mySequence">
      <xsd:sequence>
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:sequence>
</xsd:group>
```

Cette **définition de groupe modèle** est mappée dans l'allocation de type ASN.1 suivante:

```
MySequence ::= [UNTAGGED] SEQUENCE {
      a XSD.String,
      b BOOLEAN }
```

C.3.6.2 Ce qui suit est une **définition de groupe modèle** dont le **groupe modèle** a un **composeur** de **all:**

```
<xsd:group name="myAll ">
      <xsd:all>
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:all>
</xsd: group>
```

Cette **définition de groupe modèle** n'est pas mappée en ASN.1. Voir le § C.3.8.3.1 pour un exemple du mappage d'une **définition de type complexe** où le **groupe modèle** de cette **définition de groupe modèle** survient comme le **groupe modèle** supérieur.

C.3.6.3 Ce qui suit est une **définition de groupe modèle** dont le **groupe modèle** a un **composeur** de **choice**.

```
<xsd:group name="myChoice">
      <xsd:choice>
            <xsd:element name="am" type="xsd:string"/>
            <xsd:element name="bm" type="xsd:boolean"/>
      </xsd:choice>
</xsd:group>
```

Cette **définition de groupe modèle** est mappée dans l'allocation de type ASN.1 suivante:

```
MyChoice ::= [UNTAGGED] CHOICE {
      am XSD.String,
      bm BOOLEAN }
```

C.3.7 Mappage des particules

La **définition de groupe modèle** du § C.3.6.3 et ses types ASN.1 correspondants, est utilisée dans certains des exemples de **particule**.

C.3.7.1 L'exemple suivant montre les **particule**s d'un **groupe modèle** avec un **composeur** de **sequence**.

```
<xsd:complexType name="ElementSequence">
      <xsd:sequence>
            <xsd:element name="elem1" type="xsd:boolean"/>
            <xsd:element name="elem2" type="xsd:boolean" minOccurs="0"/>
            <xsd:element name="elem3" type="xsd:boolean" minOccurs="2" maxOccurs="5"/>
            <xsd:element name="elem4" type="xsd:boolean" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="elem5" type="xsd:boolean" minOccurs="5" maxOccurs="unbounded"/>
      <xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ModelGroupSequence">
      <xsd:sequence>
            <xsd:group ref="myChoice"/>
            <xsd:choice>
                  <xsd:element name="a" type="xsd:string"/>
                  <xsd:element name="b" type="xsd:string"/>
            </xsd:choice>
            <xsd:sequence>
                  <xsd:element name="c" type="xsd:string"/>
                  <xsd:element name="d" type="xsd:string"/>
            </xsd:sequence>
            <xsd:choice minOccurs="3" maxOccurs="12">
                  <xsd:element name="e" type="xsd:string"/>
                  <xsd:element name="f" type="xsd:string"/>
            </xsd:choice>

      </xsd:sequence>
</xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
ElementSequence ::= SEQUENCE {
      elem1             BOOLEAN,
      elem2             BOOLEAN OPTIONAL,
      elem3-list        [UNTAGGED] SEQUENCE (SIZE(2..5)) OF elem3 BOOLEAN,
      elem4-list        [UNTAGGED] SEQUENCE OF elem4 BOOLEAN,
      elem5-list        [UNTAGGED] SEQUENCE (SIZE(1..MAX)) OF elem5 BOOLEAN }

ModelGroupSequence ::= SEQUENCE {
      myChoice          MyChoice,
      choice            [UNTAGGED] CHOICE {
            a                 XSD.String,
            b                 XSD.String },
      sequence          [UNTAGGED] SEQUENCE {
            c                 XSD.String,
            d                 XSD.String },
      choice-list       [UNTAGGED] SEQUENCE (SIZE(3..12)) OF [UNTAGGED] CHOICE {
            e                 XSD.String,
            f                 XSD.String } }
```

C.3.7.2 Les exemples suivants montrent les **particule**s d'un **groupe modèle** avec un **composeur** de **all**.

```
<xsd:complexType name="ElementAll">
      <xsd:all>
            <xsd:element name="elem1" type="xsd:boolean"/>
            <xsd:element name="elem2" type="xsd:boolean" minOccurs="0"/>
      </xsd:all>
</xsd:complexType>
```

Cette **définition de type complexe** est mappée dans les allocations de type ASN.1 suivantes:

```
ElementAll ::= [USE-ORDER] SEQUENCE {
      order SEQUENCE OF ENUMERATED {elem1, elem2},
      elem1 XSD.String,
      elem2 XSD.String OPTIONAL }
      (CONSTRAINED BY
            {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 35 */})
```

C.3.7.3 L'exemple suivant montre les **particule**s d'un **groupe modèle** avec un **composeur** de **choice**.

```
<xsd:complexType name="ElementSequence">
	<xsd:choice>
		<xsd:element name="elem1" type="xsd:boolean"/>
		<xsd:element name="elem2" type="xsd:boolean" minOccurs="0"/>
		<xsd:element name="elem3" type="xsd:boolean" minOccurs="2" maxOccurs="5"/>
		<xsd:element name="elem4" type="xsd:boolean" minOccurs="0" maxOccurs="unbounded"/>
		<xsd:element name="elem5" type="xsd:boolean" minOccurs="5" maxOccurs="unbounded"/>
	<xsd:choice>
</xsd:complexType>

<xsd:complexType name="ModelGroupChoice">
	<xsd:choice>
		<xsd:group ref="myChoice"/>
		<xsd:choice>
			<xsd:element name="a" type="xsd:string"/>
			<xsd:element name="b" type="xsd:string"/>
		</xsd:choice>
		<xsd:sequence>
			<xsd:element name="c" type="xsd:string"/>
			<xsd:element name="d" type="xsd:string"/>
		</xsd:sequence>
		<xsd:choice minOccurs="3" maxOccurs="12">
			<xsd:element name="e" type="xsd:string"/>
			<xsd:element name="f" type="xsd:string"/>
		</xsd:choice>
	</xsd:choice>
</xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
ElementSequence ::= SEQUENCE {
      choice [UNTAGGED] CHOICE {
            elem1       BOOLEAN,
            elem2-list [UNTAGGED] SEQUENCE (SIZE(0..1)) OF elem2 BOOLEAN,
            elem3-list [UNTAGGED] SEQUENCE (SIZE(2..5)) OF elem3 BOOLEAN,
            elem4-list [UNTAGGED] SEQUENCE OF elem4 BOOLEAN,
            elem5-list [UNTAGGED] SEQUENCE (SIZE(5..MAX)) OF elem5 BOOLEAN } }

ModelGroupChoice ::= SEQUENCE {
      choice [UNTAGGED] CHOICE {
            myChoice    MyChoice,
            choice      [UNTAGGED] CHOICE {
                  a           XSD.String,
                  b           XSD.String },
            sequence    [UNTAGGED] SEQUENCE {
                  c           XSD.String,
                  d           XSD.String }
            choice-list [UNTAGGED] SEQUENCE (SIZE(3..12)) OF [UNTAGGED] CHOICE {
                  e           XSD.String,
                  f           XSD.String } }
```

C.3.8 Mappage des définitions de type complexe

C.3.8.1 L'exemple suivant est une **définition de type complexe** dont le **type de contenu** est **vide**.

```
<xsd:complexType name="Null"/>

<xsd:complexType name="Ack">
	<xsd:sequence/>
	<xsd:attribute name="packetNumber" type="xsd:integer"/>
</xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
Null ::= SEQUENCE {}

Ack ::= SEQUENCE {
      packetNumber [ATTRIBUTE] INTEGER OPTIONAL }
```

C.3.8.2 L'exemple suivant est une **définition de type complexe** dont le **type de contenu** est une **définition de type simple**.

```
<xsd:complexType name="Formatted">
	<xsd:simpleContent>
		<xsd:extension base="xsd:token">
			<xsd:attribute name="format">
				<xsd:simpleType>
					<xsd:restriction base="xsd:token">
						<xsd:enumeration value="bold"/>
```

```
                                        <xsd:enumeration value="italic"/>
                                    </xsd:restriction>
                                </xsd:simpleType>
                            </xsd:attribute>
                        </xsd:extension>
                    </xsd:simpleContent>
                </xsd:complexType>
```

Cette **définition de type complexe** est mappée dans l'allocation de type ASN.1 suivante:

```
Formatted ::= SEQUENCE {
      Format       [ATTRIBUTE] ENUMERATED {bold, italic} OPTIONAL,
      Base         [UNTAGGED] XSD.Token }
```

C.3.8.3 Les exemples suivants sont des **définitions de type complexe** dont le **type de contenu** est un modèle de contenu **element-only**.

C.3.8.3.1 Dans l'exemple suivant, le **type de contenu** est le **groupe modèle** d'une **définition de groupe modèle**.

Cet exemple utilise les types définis au § C.3.6.

```
<xsd:complexType name="MyComplexType-1">
      <xsd:group ref="myAll"/>
</xsd:complexType>

<xsd:complexType name="MyComplexType-2">
      <xsd:group ref="myChoice" />
</xsd:complexType>

<xsd:complexType name="MyComplexType-3">
      <xsd:group ref="mySequence" maxOccurs="100"/>
</xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
MyComplexType-1 ::= [USE-ORDER] SEQUENCE {
      order       SEQUENCE OF ENUMERATED {a,b},
      a           XSD.String,
      b BOOLEAN }
      (CONSTRAINED BY
            {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 35 */})

MyComplexType-2 ::= SEQUENCE {
      myChoice MyChoice }

MyComplexType-3 ::= SEQUENCE {
      mySequence-list SEQUENCE (SIZE(1..100)) OF MySequence }
```

C.3.8.3.2 Dans l'exemple suivant, le **type de contenu** est un **groupe modèle** dont le **composeur** est **choice**.

```
<xsd:complexType name="MyComplexType-4">
      <xsd:choice>
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:choice>
</xsd:complexType>

<xsd:complexType name="MyComplexType-5">
      <xsd:choice minOccurs="0">
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:choice>
</xsd:complexType>

<xsd:complexType name="MyComplexType-6">
      <xsd:choice maxOccurs="5">
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:choice>
</xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
MyComplexType-4 ::= SEQUENCE {
      choice [UNTAGGED] CHOICE {
            a XSD.String,
            b BOOLEAN } }
```

```
MyComplexType-5 ::= SEQUENCE {
      choice [UNTAGGED] CHOICE {
            a XSD.String,
            b BOOLEAN } OPTIONAL }

MyComplexType-6 ::= SEQUENCE {
      choice-list [UNTAGGED] SEQUENCE (SIZE(1..5)) OF [UNTAGGED] CHOICE {
            a XSD.String,
            b BOOLEAN } }
```

C.3.8.3.3 Dans l'exemple suivant, le **type de contenu** est un **groupe modèle** dont le **composeur** est **all**.

```
<xsd:complexType name="MyComplexType-7">
      <xsd:all>
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:all>
</xsd:complexType>

<xsd:complexType name="MyComplexType-8">
      <xsd:all minOccurs="0">
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:all>
</xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
MyComplexType-7 ::= [USE-ORDER] SEQUENCE {
      order       SEQUENCE OF ENUMERATED {a,b},
      a           XSD.String,
      b           BOOLEAN }
      (CONSTRAINED BY
            {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 35 */})

MyComplexType-8 ::= [USE-ORDER] SEQUENCE {
      order       SEQUENCE OF ENUMERATED {a,b},
      a           XSD.String OPTIONAL,
      b           BOOLEAN OPTIONAL }
      (CONSTRAINED BY
            {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 35 */})
```

C.3.8.3.4 Dans l'exemple suivant, le **type de contenu** est un **groupe modèle** dont le **composeur** est **sequence**.

```
<xsd:complexType name="MyComplexType-9">
      <xsd:sequence>
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MyComplexType-10">
      <xsd:sequence minOccurs="0">
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MyComplexType-11">
      <xsd:sequence maxOccurs="5">
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:sequence>
</xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
MyComplexType-9 ::= SEQUENCE {
      a XSD.String,
      b BOOLEAN }

MyComplexType-10 ::= SEQUENCE {
      sequence [UNTAGGED] SEQUENCE {
            a XSD.String,
            b BOOLEAN } OPTIONAL }
```

```
MyComplexType-11 ::= SEQUENCE {
      sequence-list [UNTAGGED] SEQUENCE (SIZE(1..5)) OF [UNTAGGED] SEQUENCE {
            a XSD.String,
            b BOOLEAN } }
```

C.3.8.4 L'exemple suivant montre une **définition de type complexe** dont le **type de contenu** est un modèle de contenu **mixte**.

```
<xsd:complexType name="MyComplexType-12" mixed="true">
      <xsd:sequence>
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="MyComplexType-13" mixed="true">
      <xsd:all>
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:all>
</xsd:complexType>

<xsd:complexType name="MyComplexType-14" mixed="true">
      <xsd:choice>
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:choice>
</xsd:complexType>

<xsd:complexType name="MyComplexType-15" mixed="true">
      <xsd:all minOccurs="0">
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:all>
</xsd:complexType>

<xsd:complexType name="MyComplexType-16">
      <xsd:sequence maxOccurs="unbounded" minOccurs="0">
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
      </xsd:sequence>
</xsd:complexType>
```

Ces **définition**s **de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
MyComplexType-12 ::= [EMBED-VALUES] SEQUENCE {
      embed-values      SEQUENCE OF XSD.String,
      a                 XSD.String,
      b                 BOOLEAN }
      (CONSTRAINED BY
            {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 25 */})

MyComplexType-13 ::= [EMBED-VALUES] [USE-ORDER] SEQUENCE {
      embed-values      SEQUENCE OF XSD.String,
      order             SEQUENCE OF ENUMERATED {a,b},
      a                 XSD.String,
      b                 BOOLEAN }
      (CONSTRAINED BY
            {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 25 */})
       (CONSTRAINED BY
            {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 35 */})
      MyComplexType-14 ::= [EMBED-VALUES] SEQUENCE {
      embed-values      SEQUENCE OF XSD.String,
      choice            [UNTAGGED] CHOICE {
            a                 XSD.String,
            b                 BOOLEAN } }
      (CONSTRAINED BY
            {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 25 */})

MyComplexType-15 ::= [EMBED-VALUES] [USE-ORDER] SEQUENCE {
      embed-values      SEQUENCE OF XSD.String,
      order             SEQUENCE OF ENUMERATED {a,b},
      a                 XSD.String OPTIONAL,
```

```
      b                BOOLEAN OPTIONAL }
      (CONSTRAINED BY
           {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 35 */})
       (CONSTRAINED BY
           {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 25 */})

MyComplexType-16 ::= [EMBED-VALUES] SEQUENCE {
      embed-values     SEQUENCE OF XSD.String,
      sequence-list    [UNTAGGED] SEQUENCE OF [UNTAGGED] SEQUENCE {
           a                XSD.String,
           b                BOOLEAN } }
      (CONSTRAINED BY
           {/* Doit être conforme à la Rec. UIT-T X.693 | ISO/CEI 8825-4,
      § 25 */})
```

C.3.8.5 L'exemple suivant montre les **utilisations d'attribut** d'une **définition de type complexe** construite en utilisant une **définition de groupe d'attribut**.

```
<xs:attributeGroup name="AG1">
      <xs:attribute name="a1" type="xs:string"/>
      <xs:attribute name="a2" type="xs:string"/>
      <xs:attribute name="a3" type="xs:decimal"/>
</xs:attributeGroup>

<xs:attributeGroup name="AG2">
      <xs:attribute name="a1" use="prohibited"/>
      <xs:attribute name="a3" type="xs:integer"/>
</xs:attributeGroup>

<xs:complexType name="MyComplexType-17">
      <xs:attribute name="a4" type="xs:boolean"/>
      <xs:attribute name="a5" type="xs:boolean"/>
      <xs:attributeGroup ref="AG1"/>
 </xs:complexType>

<xs:complexType name="MyComplexType-18">
      <xs:complexContent>
            <xs:restriction base="MyComplexType-17">
                  <xs:attributeGroup ref="AG2"/>
                  <xs:attribute name="a4" use="prohibited"/>
            </xs:restriction>
      </xs:complexContent>
</xs:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
MyComplexType-17 ::= SEQUENCE {
      a1 [ATTRIBUTE] XSD.String OPTIONAL,
      a2 [ATTRIBUTE] XSD.String OPTIONAL,
      a3 [ATTRIBUTE] XSD.Decimal OPTIONAL,
      a4 [ATTRIBUTE] BOOLEAN OPTIONAL,
      a5 [ATTRIBUTE] BOOLEAN OPTIONAL }

MyComplexType-18 ::= SEQUENCE {
      a2 [ATTRIBUTE] XSD.String OPTIONAL,
      a3 [ATTRIBUTE] INTEGER OPTIONAL,
      a5 [ATTRIBUTE] BOOLEAN OPTIONAL }

MyComplexType-17-derivations ::= [USE-TYPE] CHOICE {
      myComplexType-17 [NAME AS CAPITALIZED] MyComplexType-17,
      myComplexType-18 [NAME AS CAPITALIZED] MyComplexType-18 }
```

C.3.8.6 Déduction des **définitions de type complexe**

```
<xsd:complexType name="MyComplexType-19">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="a" type="xsd:string"/>
            <xsd:element name="b" type="xsd:boolean"/>
            <xsd:element name="c" type="xsd:boolean" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="attr1" type="xsd:short" use="required"/>
      <xsd:attribute name="attr2" type="xsd:short"/>
</xsd:complexType>

<xsd:complexType name="MyComplexType-20">
      <xsd:complexContent>
```

```
                <xsd:restriction base="MyComplexType-19">
                    <xsd:sequence>
                        <xsd:element name="a" type="xsd:token"/>
                        <xsd:element name="b" type="xsd:boolean"/>
                    </xsd:sequence>
                    <xsd:attribute name="attr2" type="xsd:short" use="prohibited"/>
                </xsd:restriction>
            </xsd:complexContent>
        </xsd:complexType>

        <xsd:complexType name="MyComplexType-21">
            <xsd:complexContent>
                <xsd:extension base="MyComplexType-20">
                    <xsd:sequence>
                        <xsd:element name="d" type="xsd:string"/>
                    </xsd:sequence>
                    <xsd:attribute name="attr3" type="xsd:boolean"/>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
MyComplexType-19 ::= SEQUENCE {
      attr1             [ATTRIBUTE] XSD.Short,
      attr2             [ATTRIBUTE] XSD.Short OPTIONAL,
      sequence-list     [UNTAGGED] SEQUENCE OF [UNTAGGED] SEQUENCE {
            a                 XSD.String,
            b                 BOOLEAN,
            c                 BOOLEAN OPTIONAL } }

MyComplexType-20 ::= SEQUENCE {
      attr1             [ATTRIBUTE] XSD.Short,
      a                 XSD.Token,
      b                 BOOLEAN }

MyComplexType-21 ::= SEQUENCE {
      attr1             [ATTRIBUTE] XSD.Short,
      attr3             [ATTRIBUTE] BOOLEAN OPTIONAL,
      a                 XSD.String,
      b                 BOOLEAN,
      d                 XSD.String }

MyComplexType-20-derivations ::= [USE-TYPE] CHOICE {
      myComplexType-20 [NAME AS CAPITALIZED] MyComplexType-20,
      myComplexType-21 [NAME AS CAPITALIZED] MyComplexType-21 }

MyComplexType-19-derivations ::= [USE-TYPE] CHOICE {
      myComplexType-19 [NAME AS CAPITALIZED] MyComplexType-19,
      myComplexType-20 [NAME AS CAPITALIZED] MyComplexType-20,
      myComplexType-21 [NAME AS CAPITALIZED] MyComplexType-21 }
```

C.3.9 Mappage des wildcards

Pour ces exemples, le **namespace cible** est supposé avoir l'URI suivante: "http://www.asn1.org/X694/wildcard".

C.3.9.1 **wildcard** attribut

```
        <xsd:complexType name="AnyAttribute-1">
            <xsd:anyAttribute namespace="##any"/>
        </xsd:complexType>

        <xsd:complexType name="AnyAttribute-2">
            <xsd:anyAttribute namespace="##other"/>
        </xsd:complexType>

        <xsd:complexType name="AnyAttribute-3">
            <xsd:anyAttribute namespace="##targetNamespace"/>
        </xsd:complexType>

        <xsd:complexType name="AnyAttribute-4">
            <xsd:anyAttribute namespace="##local http://www.asn1.org/X694/attribute"/>
        </xsd:complexType>

        <xsd:complexType name="AnyAttribute-5">
            <xsd:complexContent>
                <xsd:extension base="AnyAttribute-4">
                    <xsd:anyAttribute namespace="##targetNamespace"/>
                </xsd:extension>
```

```
      </xsd:complexContent>
</xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
AnyAttribute-1 ::= SEQUENCE {
      attr [ANY-ATTRIBUTES] SEQUENCE (CONSTRAINED BY {
            /* Chaque item doit être conforme au "AnyAttributeFormat" spécifié
            dans la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */})
            OF XSD.String }

AnyAttribute-2 ::= SEQUENCE {
      attr [ANY-ATTRIBUTES EXCEPT "http://www.asn1.org/X694/wildcard"]
            SEQUENCE (CONSTRAINED BY {
            /* Chaque item doit être conforme au "AnyAttributeFormat" spécifié
            dans la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */})
            OF XSD.String }

AnyAttribute-3 ::= SEQUENCE {
      attr [ANY-ATTRIBUTES FROM "http://www.asn1.org/X694/wildcard"]
            SEQUENCE (CONSTRAINED BY {
            /* Chaque item doit être conforme au "AnyAttributeFormat" spécifié
            dans la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */})
            OF XSD.String }

AnyAttribute-4 ::= SEQUENCE {
      attr [ANY-ATTRIBUTES FROM ABSENT
                                    "http://www.asn1.org/X694/attribute"]
            SEQUENCE (CONSTRAINED BY  {
            /* Chaque item doit être conforme au "AnyAttributeFormat" spécifié
            dans la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */})
            OF XSD.String }

AnyAttribute-5 ::= SEQUENCE {
      attr [ANY-ATTRIBUTES FROM ABSENT
                                    "http://www.asn1.org/X694/attribute"
                                    "http://www.asn1.org/X694/wildcard"]
            SEQUENCE (CONSTRAINED BY {
            /* Chaque item doit être conforme au "AnyAttributeFormat" spécifié
            dans la Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */})
            OF XSD.String }
```

C.3.9.2 Ce qui suit est un exemple d'une **wildcard** de modèle de contenu

```
<xsd:complexType name="Any-1">
      <xsd:sequence>
            <xsd:any namespace="##any"/>
      </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Any-2">
      <xsd:sequence>
            <xsd:any minOccurs="0" namespace="##other"/>
      </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Any-3">
      <xsd:sequence>
            <xsd:any minOccurs="0" masOccurs="unbounded" namespace="##local"/>
      </xsd:sequence>
</xsd:complexType>
```

Ces **définitions de type complexe** sont mappées dans les allocations de type ASN.1 suivantes:

```
Any-1 ::= SEQUENCE {
      elem [ANY-ELEMENT] XSD.String (CONSTRAINED BY {
                  /* Doit être conforme au "AnyElementFormat" spécifié dans la
                        Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */})}

Any-2 ::= SEQUENCE {
      elem [ANY-ELEMENT EXCEPT ABSENT
                              "http://www.asn1.org/X694/wildcard"]
            XSD.String (CONSTRAINED BY {
                  /* Doit être conforme au "AnyElementFormat" spécifié dans la
                        Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */})
                        OPTIONAL }
```

```
Any-3 ::= SEQUENCE {
      elem-list SEQUENCE OF elem
            [ANY-ELEMENT FROM ABSENT] XSD.String (CONSTRAINED BY {
                  /* Doit être conforme au "AnyElementFormat" spécifié dans la
                        Rec. UIT-T X.693 | ISO/CEI 8825-4, § 18 */})}
```

NOTE – Pour plus d'exemples sur le calcul de "NamespaceRestriction" voir les exemples sur les attributs wildcards au § C.3.9.1.

Annexe D

Utilisation du mappage pour fournir des codages binaires pour le schéma XML du W3C

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

La présente annexe décrit l'utilisation du mappage spécifié dans la présente Recommandation | Norme internationale en conjonction avec les règles de codage ASN.1 normalisées pour fournir des codages binaires canoniques et compacts pour les données définies par un schéma XSD.

D.1 Codage des schémas XSD

D.1.1 Les schémas XSD peuvent être mappés en définitions de type ASN.1 comme spécifié dans le corps de la présente Recommandation | Norme internationale et le type de niveau supérieur peut alors être codé en utilisant une des règles de codage ASN.1 spécifiées dans les Recommandations UIT-T X.690 | ISO/CEI 8825-1, UIT-T X.691 | ISO/CEI 8825-2 et UIT-T X.693 | ISO/CEI 8825-4.

D.1.2 Chacun de ces codages a une valeur d'identifiant d'objet associée qui peut être utilisée pour identifier le codage en cours de transfert. La façon selon laquelle une telle identification est communiquée à un décodeur est en dehors du domaine d'application de la présente Recommandation | Norme internationale.

D.1.3 Lorsque le schéma XSD n'est pas envoyé au destinataire par la méthode décrite au § D.3, la façon dont le destinataire obtient le schéma est en dehors du domaine d'application de la présente Recommandation | Norme internationale.

D.2 Transfert sans utilisation du schéma XSD pour les schémas

D.2.1 Cette méthode suppose que le destinataire connaît le schéma XSD utilisé par l'expéditeur.

D.2.2 La Figure D.1 montre comment utiliser le mappage défini dans la présente Recommandation | Norme internationale pour coder les documents XML au moyen des règles de codage ASN.1.

D.2.3 L'expéditeur et le destinataire utilisent le même schéma XSD (fixé) pour générer un schéma ASN.1 qui est ensuite donné à un compilateur ASN.1 pour générer le tableau de codage BER, DER, PER ou XER pour les documents XML conformes à ce schéma XSD.

D.3 Transfert utilisant le schéma XSD pour les schémas

D.3.1 Dans la mesure où un unique schéma XSD pour les schémas est disponible, il est possible de procéder en deux étapes (voir la Figure D.2).

D.3.2 L'expéditeur et le destinataire construisent un module ASN.1 et un codeur/décodeur à partir du schéma XSD pour les schémas.

D.3.3 Dans la première étape, l'expéditeur code en BER, DER ou PER le schéma XSD pour le document et envoie le schéma codé au destinataire. Le destinataire décode ce schéma et, utilisant le mappage de schéma XSD en ASN.1 et un compilateur ASN.1, génère un module ASN.1 et un codeur/décodeur pour documents XML conforme à ce schéma.

D.3.4 Dans la seconde étape, l'expéditeur code en BER, DER, PER ou XER le document XML et envoie le document codé au destinataire.

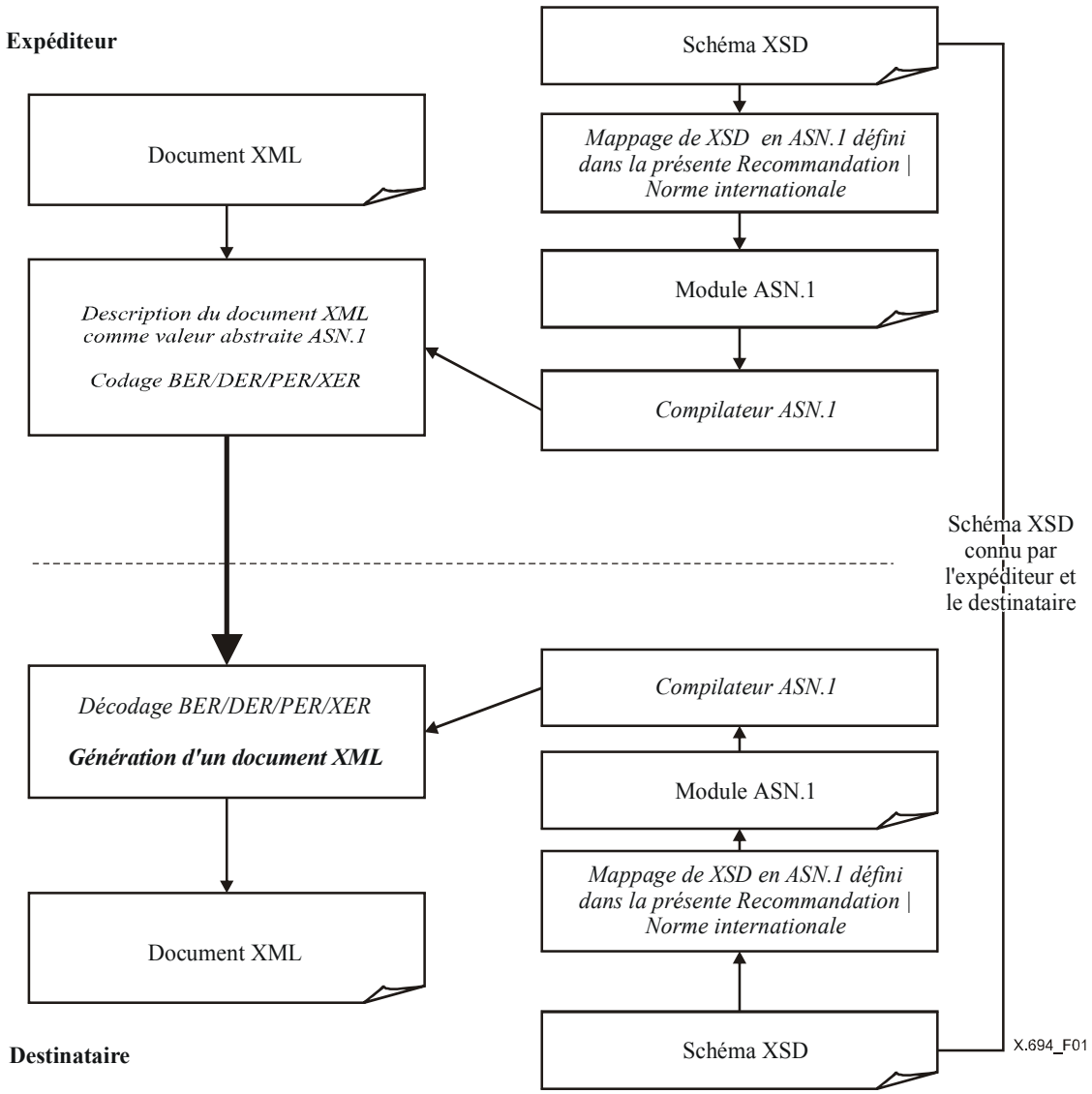

Figure D.1 – Transfert d'un document XML en utilisant le mappage de XSD en ASN.1

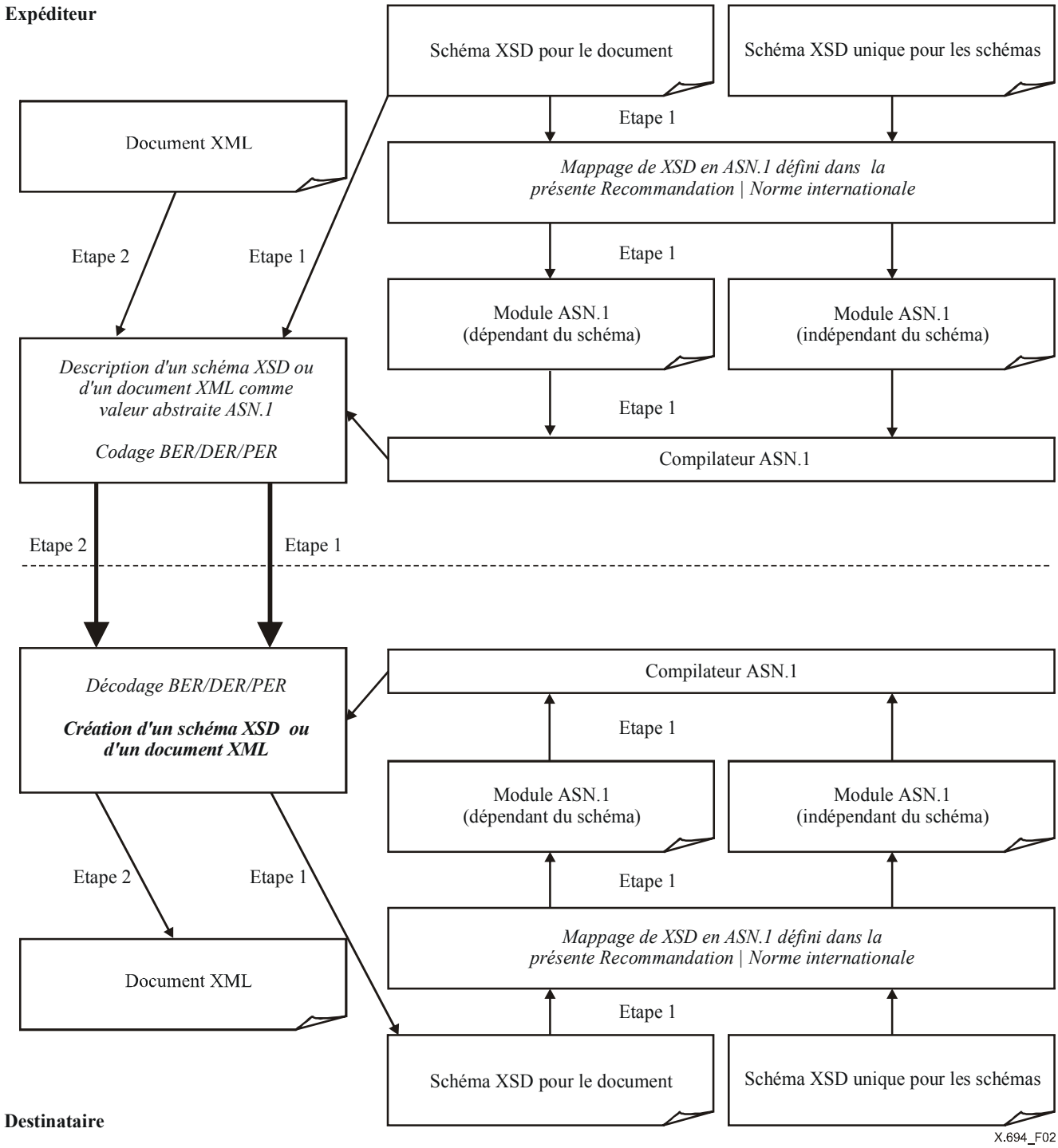

Figure D.2 – Transfert d'un shéma XSD et d'un document XML en utilisant le mappage de XSD en ASN.1

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, circuits téléphoniques, télégraphie, télécopie et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	**Réseaux de données et communication entre systèmes ouverts**
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de nouvelle génération
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Imprimé en Suisse

Genève, 2005