
**Information technology — ASN.1
encoding rules: Mapping W3C XML
schema definitions into ASN.1**

AMENDMENT 1: Efficiency enhancements

*Technologies de l'information — Règles de codage ASN.1: Mappage en
ASN.1 des définitions de schéma XML du W3C*

AMENDEMENT 1: Mises en évidence de l'efficacité



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2008

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

	Page
1) Summary	1
2) Introduction.....	1
3) Clause 1.....	2
4) Subclause 2.1.....	2
5) Subclause 2.2.....	3
6) Subclause 3.1.2	3
7) Subclause 7.1.....	3
8) Subclause 7.3.....	4
9) Subclauses 7.4, 7.5 and 7.6.....	4
10) Subclause 7.8.....	4
11) Clause 9.....	5
12) Subclause 9.1.....	5
13) Subclause 9.4.....	5
14) Subclause 9.4 <i>bis</i>	5
15) Subclause 10.1.1	5
16) Subclause 10.2.1	6
17) Subclause 10.2.2	6
18) Subclause 10.3.2	6
19) Subclause 10.3.4.1	6
20) Subclause 10.3.5	6
21) Subclause 10.4.1	7
22) Subclause 10.4.2.1	7
23) Subclauses 10.4.3, 10.4.4 and 10.4.5	7
24) Subclause 10.4.6	8
25) Clause 11 and Table 2	8
26) Subclause 12.1.1	9
27) Subclause 12.3.1	9
28) Subclause 12.4.1	10
29) Subclause 12.4.1.4.....	10
30) Subclause 12.4.3	10
31) Subclause 12.5.2.1	11
32) Subclause 13.1	11
33) Subclause 13.2	12
34) Subclause 13.3	12
35) Subclause 13.4	12
36) Subclause 13.7	12
37) Subclause 13.8	12
38) Subclauses 13.9.2 <i>bis</i> and 13.9.2 <i>ter</i>	12
39) Subclause 13.9.3	13
40) Subclause 13.10.2.....	13
41) Subclause 13.10.5.....	13
42) Subclause 14.1	13
43) Subclause 14.5	14
44) Subclause 14.6	14

45)	Subclause 18.2	15
46)	Subclause 18.3	15
47)	Subclause 18.4	15
48)	Subclause 19.1	15
49)	Subclause 19.2.1	15
50)	Subclause 19.2 <i>bis</i>	15
51)	Subclause 19.4.2 and Table 5	16
52)	Subclause 19.5	16
53)	Subclause 19.6	16
54)	Subclauses 20.1, 20.2, 20.3, 20.4 and 20.5	17
55)	Subclause 20.8	17
56)	Subclause 20.9.1	17
57)	Subclause 21.1	17
58)	Subclause 21.1 <i>bis</i>	18
59)	Subclause 21.2	18
60)	Subclause 21.2 <i>bis</i>	18
61)	Subclause 21.3	19
62)	Subclause 22.4	19
63)	Subclause 22.5	20
64)	Subclause 23.1	20
65)	Subclause 23.2	20
66)	Subclause 23.3	20
67)	Subclause 23.7	20
68)	Subclause 23.8	20
69)	Subclause 23.8.3	21
70)	Subclause 24.1	21
71)	Subclause 24.3	21
72)	Subclause 24.7	21
73)	Subclause 24.8	21
74)	Subclause 25.1	22
75)	Subclause 25.3	22
76)	Subclause 25.7	22
77)	Subclause 25.8	22
78)	Subclause 26.1	23
79)	Subclause 26.5	23
80)	Subclause 26.6	23
81)	Subclause 27.1	23
82)	Subclause 27.1 <i>bis</i>	24
83)	Subclause 27.2	24
84)	Subclause 27.7	24
85)	Subclause 27.7.1	24
86)	Subclause 27.7.1 <i>bis</i>	24
87)	Subclause 27.9	24
88)	Subclause 27.11	25
89)	Subclause 27.12	25

90)	Subclause 27.12.2	25
91)	Subclause 28.2	25
92)	Subclause 28.3	25
93)	Subclause 28.4	25
94)	Clause 29	26
95)	Clause 30	27
96)	Subclause 31.2	28
97)	Annex A	28
98)	Annex A <i>bis</i>	32
99)	Annex B	36
100)	Annex C	36

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to ISO/IEC 8825-5 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems* in collaboration with ITU-T. The identical text is published as ITU-T Rec. X.694/Amd.1 (05/2007).

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – ASN.1 encoding rules:
Mapping W3C XML schema definitions into ASN.1

Amendment 1

Efficiency enhancements

Conventions used in this amendment: Original, unchanged, text is in normal font. Deleted text is struck-through, thus: ~~deleted text~~. Inserted text is underlined, thus: inserted text. Text in new clauses is not underlined.

1) Summary

Replace the Summary with the following:

This Recommendation | International Standard defines rules for mapping an XSD Schema (a schema conforming to the W3C XML Schema specification) to an ASN.1 schema in order to use ASN.1 encoding rules such as the Basic Encoding Rules (BER), the Distinguished Encoding Rules (DER), the Packed Encoding Rules (PER) or the XML Encoding Rules (XER) for the transfer of information defined by the XSD Schema.

The use of this Recommendation | International Standard with the ASN.1 Extended XML Encoding Rules (EXTENDED-XER) provides the same XML representation of values as that defined by the original XSD Schema, but also provides the ability to encode the specified XML with an efficient binary representation (binary XML). An XML document can be converted to binary XML (for storage or transfer) using the ASN.1 generated by this mapping, and the resulting binary can be converted back to the same XML document for further XML processing.

Two versions of the mapping are defined. Version 1 of the mapping was published in 2004, and a Corrigendum was issued to this Version renaming the types **DATE-TIME** and **DURATION** in Annex A (in order to avoid conflict with the application of ITU-T Rec. X.680/Amd. 3 | ISO/IEC 8824-1/Amd. 3 (known as the time types amendment). The Version 2 mapping is more efficient in two areas: the ASN.1 time types are used rather than VisibleString for mappings of dates and times; the Fast Infoset specification (ITU-T Rec. X.891 | ISO/IEC 24824-1) is used for the mapping of XSD wild-cards. Both these changes to the mapping provide much more compact binary encodings for the XML specified by the XSD.

NOTE – The specification of the Version 1 mapping (with applicable corrections) will be maintained in the next edition of this Recommendation | International Standard, but it is expected that subsequent editions will document only the Version 2 mapping.

Application of the ASN.1 extended XML Encoding Rules to both versions of the mapping will produce the same XML (which is the same as that specified by the XSD). However, application of other ASN.1 encoding rules to the Version 1 mapping results in a verbose character-based encoding of date and time types and of XSD wild-cards, whilst application to the Version 2 mapping results in a more compact binary encoding using ASN.1 time types and the Fast Infoset specification.

2) Introduction

Replace the Introduction with the following:

This Recommendation | International Standard specifies Version 1 and Version 2 of a mapping from a W3C XML Schema definition (an XSD Schema) into an ASN.1 schema. The mappings can be applied to any XSD Schema. Both mappings specify the generation of one or more ASN.1 modules containing type definitions, together with ASN.1 XER encoding instructions. These are jointly described as an ASN.1 schema for XML documents. This ASN.1 schema (produced by either Version of the mapping), when used with the ASN.1 Extended XML Encoding Rules (EXTENDED-XER), can be used to generate and to validate the same set of W3C XML 1.0 documents as the original XSD Schema. The resulting ASN.1 types and encodings support the same semantic content as the XSD Schema.

Thus ASN.1 tools can be used interchangeably with XSD tools for the generation and processing of the specified XML documents.

Other standardized ASN.1 encoding rules, such as the Distinguished Encoding Rules (DER) or the Packed Encoding Rules (PER), can be used in conjunction with this standardized mapping, but produce encodings for Version 2 of the mapping that differ from (and are less verbose than) those produced by Version 1 for XSD constructs involving dates and times or wildcards.

The combination of this Recommendation | International Standard with ASN.1 Encoding Rules provides fully-standardized and vendor-independent compact and canonical binary encodings for data originally defined using an XSD Schema.

The ASN.1 schema provides a clear separation between the specification of the information content of messages (their abstract syntax) and the precise form of the XML document (for example, use of attributes instead of elements). This results in both a clearer and generally a less verbose schema than the original XSD Schema.

Annex A forms an integral part of this Recommendation | International Standard, and is an ASN.1 module containing a set of ASN.1 type assignments that correspond to each of the XSD built-in ~~datatypes~~ for Version 1 of the mapping. Mappings of XSD Schemas into ASN.1 schemas either import the type reference names of those type assignments or include the type definitions in-line.

Annex A bis also forms an integral part of this Recommendation | International Standard and provides the ASN.1 module for Version 2 of the mapping.

Annex B does not form an integral part of this Recommendation | International Standard, and summarizes the object identifier values assigned in this Recommendation | International Standard.

Annex C does not form an integral part of this Recommendation | International Standard, and gives examples of the mapping of XSD Schemas into ASN.1 schemas.

Annex D does not form an integral part of this Recommendation | International Standard, and describes the use of the mapping defined in this Recommendation | International Standard, in conjunction with standardized ASN.1 Encoding Rules, to provide compact and canonical encodings for data defined using an XSD Schema.

3) Clause 1

Replace the first paragraph of clause 1 with the following (retaining the remaining paragraphs):

This Recommendation | International Standard specifies two Versions of a mapping from any XSD Schema into an ASN.1 schema. The ASN.1 schema for both Versions supports the same semantics and validates the same set of XML documents.

4) Subclause 2.1

Replace subclause 2.1 with the following:

2.1 Identical Recommendations | International Standards

NOTE – The complete set of ASN.1 Recommendations | International Standards are listed below, as they can all be applicable in particular uses of this Recommendation | International Standard. Where these are not directly referenced in the body of this Recommendation | International Standard, a † symbol is added to the reference.

- ITU-T Recommendation X.680 (2002) | ISO/IEC 8824-1:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.680 (2002)/Amd. 3 (2006) | ISO/IEC 8824-1:2002/Amd. 3:2006, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation – Amendment 3: Time type support.*
- ITU-T Recommendation X.681 (2002) | ISO/IEC 8824-2:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.* †
- ITU-T Recommendation X.682 (2002) | ISO/IEC 8824-3:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (2002) | ISO/IEC 8824-4:2002, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.* †

- ITU-T Recommendation X.690 (2002) | ISO/IEC 8825-1:2002, *Information technology – ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*.
- ITU-T Recommendation X.690 (2002)/Amd.2 (2006) | ISO/IEC 8825-1:2002/Amd.2:2007, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) – Amendment 2: Time type support*.
- ITU-T Recommendation X.691 (2002) | ISO/IEC 8825-2:2002, *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*.
- ITU-T Recommendation X.691 (2002)/Amd.2 (2006) | ISO/IEC 8825-2:2002/Amd.2:2007, *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER) – Amendment 2: Time type support*. †
- ITU-T Recommendation X.692 (2002) | ISO/IEC 8825-3:2002, *Information technology – ASN.1 encoding rules: Specification of Encoding Control Notation (ECN)*. ‡
- ITU-T Recommendation X.693 (2001) | ISO/IEC 8825-4:2002, *Information technology – ASN.1 encoding rules: XML Encoding Rules (XER)*.
- ITU-T Recommendation X.693 (2001)/Amd.1 (2003) | ISO/IEC 8825-4:2002/Amd.1:2004, *Information technology – ASN.1 encoding rules: XML Encoding Rules (XER) – Amendment 1: XER encoding instructions and EXTENDED-XER*.
- ITU-T Recommendation X.693 (2002)/Amd.2 (2006) | ISO/IEC 8825-4:2002/Amd.2:2006, *Information technology – ASN.1 encoding rules: XML Encoding Rules (XER) – Amendment 2: Time type support*.
- ITU-T Recommendation X.891 (2005) | ISO/IEC 24824-1:2007, *Information technology – Generic applications of ASN.1: Fast Infoset*.

5) Subclause 2.2

In subclause 2.2, replace "ISO 8601:2000" with "ISO 8601:2004".

6) Subclause 3.1.2

Replace subclause 3.1.2 with the following:

3.1.2 This Recommendation | International Standard also uses the terms defined in W3C XML Schema and W3C XML Information Set.

NOTE 1 – It is believed that these terms do not conflict with the terms referenced in 3.1.1. If such a conflict occurs, the definition of the term in 3.1.1 applies.

NOTE 2 – In particular, the terms "schema component" and "property (of a schema component)" are defined in W3C XML Schema, and the terms "element information item" and "attribute information item" ~~are~~ defined in W3C XML Information Set.

NOTE 3 – The terms "top-level **simple type definition**" and "top-level **complex type definition**" do not include XSD built-in types, when used in this Recommendation | International Standard.

7) Subclause 7.1

Replace subclause 7.1 with the following:

7.1 A mapping is based on a source XSD Schema, which is a set of schema components (see W3C XML Schema Part 1, 2.2). No particular representation of schema components or sets of schema components is required or assumed for the mapping, although it is expected that the source XSD Schema will usually be provided as one or more XML schema documents (see W3C XML Schema Part 1, 3.15.2).

NOTE 1 – The schema components represented in multiple XML schema documents become part of the same XSD Schema through the use of the **xsd:include**, **xsd:redefine** and **xsd:import** element information items.

NOTE 2 – Since the mapping is defined in terms of schema components (and not in terms of their XML representation), it is not affected by details of the XML representation, such as the use of multiple schema documents linked by **xsd:include** and **xsd:redefine** element information items, the placement of element information items in one or another schema documents, the order of **xsd:attribute** element information items within a **xsd:complexType** element information item, and so on.

NOTE 3 – Two sets of schema documents that differ in many aspects but represent the same set of schema components generate the same set of ASN.1 type assignments, with the same final encoding instructions assigned to them and to their components to any depth.

8) Subclause 7.3

Replace subclause 7.3 with the following:

7.3 ~~At least one ASN.1 module (see 7.4) shall be generated for each different **target namespace** (whether a namespace name or the keyword **absent**) that is the **target namespace** of one or more schema components in the source XSD Schema. One or more ASN.1 modules shall be generated for a source XSD Schema. The number of ASN.1 modules generated is an implementation option. Each ASN.1 module shall contain one or more type assignments corresponding to top-level schema components (see 7.9) that have the same **target namespace**. Each ASN.1 module can also contain one or more special ASN.1 type assignments whose associated ASN.1 type assignments are in the same ASN.1 module (see 7.6). Each ASN.1 module shall contain zero or more type assignments corresponding to top-level schema components (see 7.9), and zero or more special ASN.1 type assignments (see clauses 29, 30 and 31). The physical order of type assignments within each ASN.1 module is an implementation option. When multiple ASN.1 modules are generated, the way the generated type assignments are distributed across those ASN.1 modules is also an implementation option.~~

~~NOTE – The schema components represented in the multiple schema documents become part of the same XSD Schema through the use of the **xsd:include**, **xsd:redefine**, and **xsd:import** element information items.~~

~~NOTE 1 – The inclusion in the same ASN.1 module of type assignments generated from XSD schema components with different **target namespaces** is permitted by this subclause but not recommended. The preferred mapping is to generate one ASN.1 module per namespace whenever possible. It is also recommended that each special ASN.1 type assignment be inserted in the same ASN.1 module as its associated ASN.1 type assignment (see 29.4, 30.4 and 31.4).~~

~~NOTE 2 – The generation of ASN.1 type assignments (see 7.9 and 10.4) is not affected by the number of ASN.1 modules being generated (except for the possible use of "ExternalTypeReference" as specified in 10.2.2), nor by the way the generated type assignments are distributed across those modules, nor by the physical order of the type assignments within each module. In particular, the type reference names of those type assignments are the same whatever mapping style is used by the implementation.~~

~~NOTE 3 – A full description of the relationship between the namespace concept of W3C XML Namespaces and naming in ASN.1 is provided in ITU-T Rec. X.693 | ISO/IEC 8825-4, clause 16. Type reference names and identifiers defined in an ASN.1 module are assigned a namespace by means of a **NAMESPACE** encoding instruction, and otherwise do not have a namespace. The mapping generates **NAMESPACE** encoding instructions where needed.~~

9) Subclauses 7.4, 7.5 and 7.6

Delete subclauses 7.4, 7.5 and 7.6.

10) Subclause 7.8

Replace subclause 7.8 with the following:

7.8 A source XSD Schema shall be processed as follows:

- a) for each top-level **element declaration**, an ASN.1 type assignment shall be generated by applying clause 14 to the **element declaration**;
- b) for each top-level **attribute declaration**, an ASN.1 type assignment shall be generated by applying clause 15 to the **attribute declaration**;
- c) for each ~~user-defined~~ top-level **simple type definition**, an ASN.1 type assignment shall be generated by applying clause 13 to the **simple type definition**;
- d) for each top-level **complex type definition**, an ASN.1 type assignment shall be generated by applying clause 20 to the **complex type definition**;
- e) for each **model group definition** whose **model group** has a **compositor** of **sequence** or **choice**, an ASN.1 type assignment shall be generated by applying clause 17 to the **model group definition**.

NOTE 1 – The remaining schema components of the source XSD schema will be processed as a result of mapping these schema components.

NOTE 2 – The order in which schema components are to be mapped is specified in 10.4. The order of the items of the list above has no significance for the mapping.

11) Clause 9

Replace clause 9 with the following:

9 The ASN.1 modules and namespaces

~~NOTE — A full description of the relationship between the namespace concept of W3C XML Namespaces and naming in ASN.1 is provided in ITU-T Rec. X.693 | ISO/IEC 8825-4, clause 16. Type reference names and identifiers defined in an ASN.1 module are assigned a namespace by means of a **NAMESPACE** encoding instruction, and otherwise do not have a namespace. The mapping generates **NAMESPACE** encoding instructions as appropriate.~~

12) Subclause 9.1

Replace subclause 9.1 with the following:

9.1 The mapping of an XSD Schema generates one or more ASN.1 modules (see 7.3) ~~corresponding to all schema components in the Schema that have the same target namespace.~~

13) Subclause 9.4

Replace subclause 9.4 with the following:

9.4 In each ASN.1 module generated by a Version 1 mapping, there shall be an ASN.1 **IMPORTS** statement importing the ASN.1 type reference names in the module named **XSD {joint-iso-itu-t asn1(1) specification(0) modules(0) xsd-module(2) version1(1)}** specified in Annex A that are referenced in the ASN.1 module.

14) Subclause 9.4 bis

Add a new subclause 9.4 bis as follows:

9.4 bis In each ASN.1 module generated by a Version 2 mapping, there shall be an ASN.1 **IMPORTS** statement importing the ASN.1 type reference names in the module named **XSD {joint-iso-itu-t asn1(1) specification(0) modules(0) xsd-module(2) version2(2)}** specified in Annex A bis that are referenced in the generated ASN.1 module.

NOTE – The term "XSD module" in this Recommendation | International Standard refers to the module defined in Annex A (Version 1 mapping) or in Annex A bis (Version 2 mapping), according to the version of the mapping.

15) Subclause 10.1.1

Replace subclause 10.1.1 with the following:

10.1.1 This Recommendation | International Standard specifies the generation of:

- a) ASN.1 type reference names corresponding to the **names of model group definitions**, top-level **element declarations**, top-level **attribute declarations**, top-level **complex type definitions**, and ~~user-defined~~ top-level **simple type definitions**;
- b) ASN.1 identifiers corresponding to the **names of top-level element declarations**, top-level **attribute declarations**, local **element declarations**, and local **attribute declarations**;
- c) ASN.1 identifiers for the mapping of certain **simple type definitions** with an **enumeration** facet (see 12.4.1 and 12.4.2);
- d) ASN.1 type reference names of special type assignments (see clauses 29, 30 and 31); and
- e) ASN.1 identifiers of certain sequence components introduced by the mapping (see clause 20).

16) Subclause 10.2.1

Replace subclause 10.2.1 with the following:

10.2.1 This subclause applies as explicitly invoked by other clauses of this Recommendation | International Standard to generate an ASN.1 type (a "DefinedType") definition that is a reference (a "DefinedType") to an ASN.1 type assignment.

17) Subclause 10.2.2

Replace subclause 10.2.2 with the following:

10.2.2 If an ASN.1 type definition (R, say) that is a "DefinedType" is to be inserted in an ASN.1 module (M, say) other than the ASN.1 module where the referenced ASN.1 type assignment (TA, say) is being inserted, and the type reference name of TA is identical to either the type reference name of another ASN.1 type assignment being inserted in module M or to another type reference name being imported into module M, then R the "DefinedType" shall be an "ExternalTypeReference" (constructed as appropriate for module M) for TA; for that type assignment, as an implementation option. Otherwise, it shall be a "typereference" for TA that type assignment.

NOTE – All ASN.1 "typereference"s created by the mapping are unique for any legal input schema, so a type defined in another ASN.1 module does not need to be an "ExternalTypeReference".

18) Subclause 10.3.2

Replace subclause 10.3.2 with the following:

10.3.2 Names of attribute declarations, element declarations, model group definitions, user-defined top-level simple type definitions, and top-level complex type definitions can be identical to ASN.1 reserved words or can contain characters not allowed in ASN.1 identifiers or in ASN.1 type reference names. In addition, there are cases in which ASN.1 names are required to be distinct where the names of the corresponding XSD schema components (from which the ASN.1 names are mapped) are allowed to be identical.

19) Subclause 10.3.4.1

Replace subclause 10.3.4.1 with the following:

10.3.4.1 If the name being generated is the type reference name of an ASN.1 type assignment and the character string generated by 10.3.3 is identical to:

- a) the type reference name of another ASN.1 type assignment previously (see 10.4) generated by the mapping (in any ASN.1 module); in the same ASN.1 module or in another ASN.1 module with the same namespace (including absence of a namespace); or
- b) the type reference name of a type assignment in the XSD module (see Annex A); or
- c) is one of the reserved words specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.27,

then a suffix shall be appended to the character string generated by 10.3.3. The suffix shall consist of a HYPHEN-MINUS followed by the canonical lexical representation (see W3C XML Schema Part 2, 2.3.1) of an integer. This integer shall be the least positive integer such that the new name is different from the type reference name of any other ASN.1 type assignment previously generated (in any ASN.1 module) in any of those ASN.1 modules.

NOTE – As a consequence of this rule, all type reference names defined in an ASN.1 specification generated from a source XSD schema (including the standardized type references defined in the XSD module) will be unique within that ASN.1 specification. This allows maximum flexibility in the way that the generated ASN.1 type assignments are distributed across multiple ASN.1 modules (see 7.3).

20) Subclause 10.3.5

Replace subclause 10.3.5 with the following:

10.3.5 For an ASN.1 type reference name (or identifier) that is generated by applying this subclause 10.3 to the name of an element declaration, attribute declaration, top-level complex type definition or user-defined top-level simple type definition, if the type reference name (or identifier) generated is different from the name, a final NAME encoding instruction shall be assigned to the ASN.1 type assignment with that type reference name (or to the component with that identifier) as specified in the three following subclauses.

21) Subclause 10.4.1

Replace subclause 10.4.1 with the following:

10.4.1 An order is imposed on the top-level schema components of the source XSD Schema on which the mapping is performed. This applies to **model group definitions**, top-level **complex type definitions**, ~~user defined top-level simple type definitions~~, top-level **attribute declarations**, and top-level **element declarations**.

NOTE – Other top-level schema components are not mapped to ASN.1, and XSD built-in datatypes are mapped in a special way.

22) Subclause 10.4.2.1

Replace subclause 10.4.2.1 with the following:

10.4.2.1 Top-level schema components shall first be ordered by their **target namespace**, with the **absent** namespace preceding all namespace names specified in the XSD schema in ascending lexicographical order.

23) Subclauses 10.4.3, 10.4.4 and 10.4.5

Replace subclauses 10.4.3, 10.4.4 and 10.4.5 with the following:

10.4.3 ~~The mapping generates~~ Two sets of ASN.1 type assignments are generated by the mapping:

- a) one set of ASN.1 type assignments (generated by clauses 13, 14, 15, 17 and 20) correspond directly to top-level schema components, and their type reference names are derived from the name of the schema component with no suffix appended;
- b) another set of ASN.1 type assignments (generated by clauses 29, 30 and 31) correspond to special uses of top-level schema components, and their type reference names are generated from the name of the schema component followed by a suffix and (in some cases) by a post-suffix.

NOTE – For each top-level schema component in the source XSD Schema, at most one ASN.1 type assignment in the set in 10.4.3 (a) can be generated, but multiple ASN.1 type assignments in the set in 10.4.3 (b) can be generated.

some ASN.1 type assignments that do not correspond directly to any XSD schema component. These are:

- a) ~~choice types (with a final USE TYPE encoding instruction) corresponding to a type derivation hierarchy; the type reference names of these types have a "derivations" suffix (see clause 29);~~
- b) ~~choice types (with a final USE TYPE encoding instruction on the type and a final USE NIL encoding instruction on each alternative) corresponding to a type derivation hierarchy where the user defined top-level simple type definition or complex type definition that is the root of the derivation hierarchy is used as the type definition of one or more element declarations that are nillable; the type reference names of these types have a "deriv nillable" suffix (see clause 29);~~
- e) ~~choice types (with a final USE TYPE encoding instruction on the type and a final DEFAULT FOR EMPTY encoding instruction on each alternative) corresponding to a type derivation hierarchy where the user defined top-level simple type definition or complex type definition that is the root of the derivation hierarchy is used as the type definition of one or more element declarations that are not nillable and have a value constraint that is a default value; the type reference names of these types have a "deriv default" suffix (see clause 29);~~
- d) ~~choice types (with a final USE TYPE encoding instruction on the type and a final DEFAULT FOR EMPTY encoding instruction on each alternative) corresponding to a type derivation hierarchy where the user defined top-level simple type definition or complex type definition that is the root of the derivation hierarchy is used as the type definition of one or more element declarations that are not nillable and have a value constraint that is a fixed value; the type reference names of these types have a "deriv fixed" suffix (see clause 29);~~
- e) ~~choice types (with a final USE TYPE encoding instruction on the type and final USE NIL and DEFAULT FOR EMPTY encoding instructions on each alternative) corresponding to a type derivation hierarchy where the user defined top-level simple type definition or complex type definition that is the root of the derivation hierarchy is used as the type definition of one or more element declarations that are nillable and have a value constraint that is a default value; the type reference names of these types have a "deriv nillable default" suffix (see clause 29);~~
- f) ~~choice types (with a final USE TYPE encoding instruction on the type and final USE NIL and DEFAULT FOR EMPTY encoding instructions on each alternative) corresponding to a type derivation hierarchy where the user defined top-level simple type definition or complex type definition that is the root of the~~

~~derivation hierarchy is used as the **type definition** of one or more **element declarations** that are **nillable** and have a **value constraint** that is a **fixed** value; the type reference names of these types have a "**deriv-nillable-fixed**" suffix (see clause 29);~~

- ~~g) choice types (with a final **UNTAGGED** encoding instruction) corresponding to an element substitution group; the type reference names of these types have a "**group**" suffix (see clause 31);~~
- ~~h) sequence types (with a final **USE NIL** encoding instruction) corresponding to the use of a user defined top level **simple type definition** or **complex type definition** as the **type definition** of one or more **element declarations** that are **nillable**; the type reference names of these types have a "**nillable**" suffix (see clause 30).~~

~~**10.4.4** All ASN.1 type assignments that correspond directly to the XSD schema components in the source XSD Schema shall be generated before all ASN.1 type assignments listed in 10.4.3 (if any).~~

~~**10.4.4-5** ASN.1 type assignments that correspond directly to the XSD schema components in the set in 10.4.3 (a) shall be generated in the order of the corresponding XSD schema components (see 10.4.1), and shall all be generated before any type assignments in 10.4.3 (b) are generated.~~

~~**10.4.5** ASN.1 type assignments listed in 10.4.3 (if any) shall be generated in the following order:~~

- ~~a) given two top-level schema components SC1 and SC2, where SC1 precedes SC2 in the order specified in 10.4.1, all the ASN.1 type assignments corresponding to SC1 (if any) shall be generated before any type assignments corresponding to SC2 are generated;~~
- ~~b) within each set of type assignments corresponding to any given schema component, type assignments shall be generated in an order based on the suffix specified in clauses 29 to 31, as follows:

 - ~~1) suffix "**-nillable**";~~
 - ~~2) suffix "**-nillable-default**";~~
 - ~~3) suffix "**-nillable-fixed**";~~
 - ~~4) suffix "**-derivations**";~~
 - ~~5) suffix "**-deriv-default**";~~
 - ~~6) suffix "**-deriv-fixed**";~~
 - ~~7) suffix "**-deriv-nillable**";~~
 - ~~8) suffix "**-deriv-nillable-default**";~~
 - ~~9) suffix "**-deriv-nillable-fixed**";~~
 - ~~10) suffix "**-group**";~~~~
- ~~c) for items 2, 3, 5, 6, 8 and 9 of (b), within each set of type assignments corresponding to any given schema component and any given suffix, type assignments shall be generated in ascending lexicographical order of the post-suffix specified in clause 29 (if any).~~

~~order of the XSD schema components (see 10.4.1) corresponding to the "associated type assignment" (see clauses 29, 30, and 31).~~

24) Subclause 10.4.6

Delete subclause 10.4.6.

25) Clause 11 and Table 2

Replace clause 11 and Table 2 with the following:

11 Mapping uses of XSD built-in datatypes

11.1 This clause applies as explicitly invoked by other clauses of this Recommendation | International Standard to generate an ASN.1 type definition corresponding to the use of an XSD built-in ~~data~~ type.

NOTE – All XSD built-in types are **simple type definitions** with the exception of **xsd:anyType**, which is a **complex type definition**.

11.2 A use of an XSD built-in ~~data~~ type shall be mapped to an ASN.1 type definition in accordance with Table 2. The table gives the ASN.1 type definition to be used. The notation "XSD.Name" indicates that the ASN.1 type

definition shall be the ASN.1 type definition (a "DefinedType") generated by applying 10.2 to the corresponding ASN.1 type assignment present in the XSD {joint-iso-itu-t asn1(1) specification(0) modules(0) xsd-module(2) version1(1)} module (Version 1 mapping – see Annex A) or the XSD {joint-iso-itu-t asn1(1) specification(0) modules(0) xsd-module(2) version2(2)} module (Version 2 mapping – see Annex A bis).

Table 2 – ASN.1 type definitions corresponding to uses of XSD built-in datatypes

XSD built-in datatype	ASN.1 type definition	XSD built-in datatype	ASN.1 type definition
anyURI	XSD.AnyURI	int	XSD.Int
anySimpleType	XSD.AnySimpleType	integer	INTEGER
anyType	XSD.AnyType or <u>XSD.AnyType-nillable</u> (see 11.3)	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

11.3 A use of `xsd:anyType` as the **type definition** of an **element declaration** that is not nillable shall be mapped to `XSD.AnyType`. A use of `xsd:anyType` as the **type definition** of an **element declaration** that is nillable shall be mapped to `XSD.AnyType-nillable`.

26) Subclause 12.1.1

Replace subclause 12.1.1 with the following:

12.1.1 The **length**, **minLength**, and **maxLength** facets shall be ignored for the XSD built-in datatypes `xsd:QName` and `xsd:NOTATION` and for any **simple type definition** derived from these by restriction.

27) Subclause 12.3.1

Replace subclause 12.3.1 with the following:

12.3.1 If a **whiteSpace** facet with a **value** of **replace** or **collapse** belongs to a **simple type definition** that has also an **enumeration** facet being mapped to an ASN.1 "Enumeration" (see 12.4.1 and 12.4.2), then the three following subclauses apply.

12.3.1.1 ~~¶~~No "EnumerationItem"s shall be included in the "Enumeration" for the members (if any) of the **value** of the **enumeration** facet that contain any of the characters HORIZONTAL TABULATION, NEWLINE or CARRIAGE RETURN, or (in the case of **collapse**) contain leading, trailing, or multiple consecutive SPACE characters.

12.3.1.2 If the **value** of the **whiteSpace** facet is **replace** and a final **TEXT** encoding instruction with qualifying information is being assigned to the ASN.1 type definition, then a final **WHITESPACE REPLACE** encoding instruction shall also be assigned to it.

12.3.1.3 If the **value** of the **whiteSpace** facet is **collapse** and a final **TEXT** encoding instruction with qualifying information is being assigned to the ASN.1 type definition, then a final **WHITESPACE COLLAPSE** encoding instruction shall also be assigned to it.

28) Subclause 12.4.1

Replace subclause 12.4.1 with the following:

12.4.1 An **enumeration** facet belonging to a **simple type definition** with a **variety** of **atomic** that is derived by restriction (directly or indirectly) from **xsd:string** shall not be mapped to an ASN.1 constraint. Instead, the facet shall be mapped to the "Enumeration" of the ASN.1 enumerated type corresponding to the **simple type definition** (see 13.5) as specified in the ~~three~~**four** following subclauses.

29) Subclause 12.4.1.4

Add a new subclause 12.4.1.4 as follows:

12.4.1.4 If the **simple type definition** has a **whiteSpace** facet with the value **preserve** or **replace**, then the enumerated type shall be assigned at least one final **TEXT** encoding instruction with qualifying information indicating one or more of the "EnumerationItem"s.

NOTE – An important example of this is a restriction of **xsd:string** with an **enumeration** facet, which has **whiteSpace preserve** by default.

30) Subclause 12.4.3

Replace subclause 12.4.3 with the following:

12.4.3 Any other **enumeration** facet shall be mapped to an ASN.1 constraint that is either a single value or a union of single values corresponding to the members of the **value** of the **enumeration**.

NOTE – The **enumeration** facet applies to the value space of the **base type definition**. Therefore, for an **enumeration** of the XSD built-in datatypes **xsd:QName** or **xsd:NOTATION**, the value of the **uri** component of the **[USE-QNAME] SEQUENCE** produced as a single value ASN.1 constraint is determined, in the XML representation of an XSD Schema, by the namespace declarations whose scope includes the **xsd:QName** or **xsd:NOTATION**, and by the prefix (if any) of the **xsd:QName** or **xsd:NOTATION**.

EXAMPLE 1 – The following represents a ~~user-defined~~ top-level **simple type definition** that is a restriction of **xsd:string** with an **enumeration** facet:

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

It is mapped to the ASN.1 type assignment:

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

EXAMPLE 2 – The following represents a ~~user-defined~~ top-level **simple type definition** that is a restriction of **xsd:integer** with an **enumeration** facet:

```
<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>
```

It is mapped to the ASN.1 type assignment:

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

EXAMPLE 3 – The following represents a ~~user-defined~~ top-level **simple type definition** that is a restriction of **xsd:integer** with a **minInclusive** and a **maxInclusive** facet:

```
<xsd:simpleType name="integer-1-10">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="10"/>
  </xsd:restriction>
</xsd:simpleType>
```

It is mapped to the ASN.1 type assignment:

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

EXAMPLE 4 – The following represents a ~~user-defined~~ top-level **simple type definition** that is a restriction (with a **minExclusive** facet) of another **simple type definition**, derived by restriction from **xsd:integer** with the addition of a **minInclusive** and a **maxInclusive** facet:

```
<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>
```

It is mapped to the ASN.1 type assignment:

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

EXAMPLE 5 – The following represents a ~~user-defined~~ top-level **simple type definition** that is a restriction (with a **minLength** and a **maxLength** facet) of another **simple type definition**, derived by restriction from **xsd:string** with the addition of an **enumeration** facet:

```
<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>
```

It is mapped to the ASN.1 type assignment:

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

31) Subclause 12.5.2.1

Replace subclause 12.5.2.1 with the following:

12.5.2.1 If the **simple type definition** is derived by restriction (directly or indirectly) from an XSD built-in date or time datatype (**xsd:date**, **xsd:dateTime**, **xsd:duration**, **xsd:gDay**, **xsd:gMonth**, **xsd:gYear**, **xsd:gYearMonth**, **xsd:gMonthDay**, or **xsd:time**), then the **maxInclusive**, **maxExclusive**, **minExclusive**, and **minInclusive** facets of the **simple type definition** shall be mapped to an ASN.1 user-defined constraint (see 12.5.4).

32) Subclause 13.1

Replace subclause 13.1 with the following:

13.1 This clause applies as explicitly invoked by other clauses of this Recommendation | International Standard to generate an ASN.1 type assignment or ASN.1 type definition corresponding to a **simple type definition**.

NOTE – This clause is not invoked for **simple type definitions** that are XSD built-in types.

33) Subclause 13.2

Delete subclause 13.2.

34) Subclause 13.3

Replace subclause 13.3 with the following:

13.3 A ~~user-defined~~ top-level **simple type definition** shall be mapped to an ASN.1 type assignment. The "typereference" in the "TypeAssignment" shall be generated by applying 10.3 to the **name** of the **simple type definition** and the "Type" in the "TypeAssignment" shall be an ASN.1 type definition as specified in subclauses 13.5 to 13.10.

35) Subclause 13.4

Replace subclause 13.4 with the following:

13.4 An anonymous **simple type definition** shall be mapped to an ASN.1 type definition as specified in subclauses 13.5 to 13.10.

36) Subclause 13.7

Replace subclause 13.7 with the following:

13.7 For any other **simple type definition** (D, say) with any **variety** that is derived by restriction (directly or indirectly) from a ~~user-defined~~ top-level **simple type definition**, the ASN.1 type definition shall be generated by applying clause 23 to the ~~user-defined~~ top-level **simple type definition** (B, say) such that:

- a) D is derived by restriction (directly or indirectly) from B; and
- b) either B is the **base type definition** of D, or all intermediate derivation steps from B to D are anonymous **simple type definitions**.

Then, for each of the **facets** of D (if any), an ASN.1 constraint generated by applying clause 12 to the facet shall be added to the ASN.1 type definition.

37) Subclause 13.8

Replace subclause 13.8 with the following:

13.8 For any other **simple type definition** (D, say) with a **variety** of **atomic**, the ASN.1 type definition shall be generated by applying clause 23 to the XSD built-in ~~data~~ type (B, say) such that:

- a) D is derived by restriction (directly or indirectly) from B; and
- b) either B is the **base type definition** of D, or all intermediate derivation steps from B to D are anonymous **simple type definitions**.

Then, for each of the **facets** of D, an ASN.1 constraint generated by applying clause 12 to the facet shall be added to the ASN.1 type definition.

38) Subclauses 13.9.2 bis and 13.9.2 ter

Add new subclauses 13.9.2 bis and 13.9.2 ter as follows:

13.9.2 bis If the **item type definition** of the list is **xsd:string** or a restriction of **xsd:string** and is mapped to an ASN.1 character string type, then the permitted alphabet constraint (**FROM**((0, 0, 0, 33) .. (0, 16, 255, 253))) shall be applied to the ASN.1 character string type.

13.9.2 ter If the **item type definition** of the list is a union type, then the subtype constraint specified in 13.9.2 bis shall be applied to each alternative of the ASN.1 choice type that is a character string type by using an inner subtype constraint applied to the choice type.

39) Subclause 13.9.3

Replace subclause 13.9.3 with the following:

13.9.3 A final **LIST** encoding instruction shall be assigned to the ASN.1 sequence-of type.

EXAMPLE – The following represents a ~~user-defined~~ top-level **simple type definition** that is a **list** of **xsd:float**:

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

It is mapped to the ASN.1 type assignment:

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

40) Subclause 13.10.2

Replace subclause 13.10.2 with the following:

13.10.2 For each member of the **member type definitions**, the "identifier" in the "NamedType" of the corresponding alternative shall be generated by applying 10.3 either to the **name** of the member (if the member is an XSD built-in **datatype** or a ~~user-defined~~ top-level **simple type definition**) or to the character string "**alt**" (if the member is an anonymous **simple type definition**), and the "Type" in the "NamedType" shall be the ASN.1 type definition generated by applying clause 23 to the member of the **member type definitions**.

41) Subclause 13.10.5

Replace subclause 13.10.5 with the following:

13.10.5 A final **USE-UNION** encoding instruction shall be assigned to the ASN.1 choice type.

EXAMPLE – The following represents a ~~user-defined~~ top-level **simple type definition** that is a **union** of two anonymous **simple type definitions**:

```
<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>
```

It is mapped to the ASN.1 type assignment:

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

42) Subclause 14.1

Replace subclause 14.1 with the following:

14.1 This clause applies as explicitly invoked by other clauses of this Recommendation | International Standard to generate an ASN.1 type assignment or ASN.1 type definition corresponding to an **element declaration**.

NOTE – The presence of a **value constraint** on an **element declaration** normally affects the mapping. However, 8.10 implies that an **element declaration** that has a **value constraint** and whose **type definition** is **xsd:QName** or **xsd:NOTATION** or a restriction of these XSD built-in ~~datatypes~~ is mapped as if it had no **value constraint**.

43) Subclause 14.5

Replace subclause 14.5 with the following:

14.5 One of the two following subclauses (14.5.1 and 14.5.2) applies. The ASN.1 type definition shall be generated either by applying clause 23, clause 26 or clause 27 (see 14.5 bis) to the **simple or complex type definition** that is the **type definition** of the **element declaration**, or by applying 10.2 to the ASN.1 type assignment generated by applying clause 29 to the **type definition**. In both cases, the **value constraint** in the **element declaration** (if any) shall be provided to the applicable clause (23, 26, 27 or 29) and shall be used when generating the ASN.1 type definition as specified in that clause.

14.5 bis The applicable clause number shall be obtained from the last column of Table 4 bis after selecting a row of the table based on the following conditions:

- a) whether the **element declaration** has a substitutable or a non-substitutable **type definition** (see 14.6);
- b) whether the **element declaration** is **nullable** or non-nullable;
- c) whether the **type definition** is a **simple type definition** or a **complex type definition**; and
- d) whether the **type definition** is an XSD built-in, anonymous, or top-level type definition.

Table 4 bis – Applicable clause numbers for the mapping of element declarations

Substitutable	Nullable	Simple/Complex	Type definition	Applicable clause number
<u>no</u>	<u>no</u>	<u>simple or complex</u>	<u>XSD built-in, anonymous, or top-level</u>	<u>23</u>
<u>no</u>	<u>yes</u>	<u>simple</u>	<u>XSD built-in or anonymous</u>	<u>26</u>
<u>no</u>	<u>yes</u>	<u>simple</u>	<u>top-level</u>	<u>29</u>
<u>no</u>	<u>yes</u>	<u>complex</u>	<u>XSD built-in or anonymous</u>	<u>27</u>
<u>no</u>	<u>yes</u>	<u>complex</u>	<u>top-level</u>	<u>29</u>
<u>yes</u>	<u>yes or no</u>	<u>simple or complex</u>	<u>XSD built-in, anonymous, or top-level</u>	<u>29</u>

14.5.1 If the **type definition** of the **element declaration** is an anonymous **simple type definition** or **complex type definition** or an XSD built-in datatype (A, say), then one of the two following subclauses applies.

14.5.1.1 If the **element declaration** is not **nullable**, then the ASN.1 type definition shall be generated by applying clause 23 to A.

14.5.1.2 If the **element declaration** is **nullable**, then the ASN.1 type definition shall be generated by applying either clause 26 (if A is a **simple type definition**) or clause 27 (if A is a **complex type definition**) to A.

14.5.2 If the **type definition** of the **element declaration** is a user-defined top-level **simple type definition** or **complex type definition** (T, say), then one of the four following subclauses applies.

14.5.2.1 If the **element declaration** is not **nullable** and does not have a substitutable **type definition** (see 14.6), then the ASN.1 type definition shall be generated by applying clause 23 to T.

14.5.2.2 If the **element declaration** is **nullable** and does not have a substitutable **type definition** (see 14.6), then the ASN.1 type definition shall be generated by applying either clause 26 (if T is a **simple type definition**) or clause 27 (if T is a **complex type definition**) to T.

14.5.2.3 If the **element declaration** is not **nullable** and has a substitutable **type definition** (see 14.6), then the ASN.1 type definition shall be generated by applying clause 24 to T.

14.5.2.4 If the **element declaration** is **nullable** and has a substitutable **type definition** (see 14.6), then the ASN.1 type definition shall be generated by applying clause 25 to T.

44) Subclause 14.6

Replace subclause 14.6 with the following:

14.6 The phrase "has a substitutable **type definition**", applied to an **element declaration**, means that the **type definition** of the **element declaration** is a user-defined top-level **simple type definition** or **complex type definition** that occurs as the **base type definition** of another top-level **simple type definition** or **complex type definition**.