INTERNATIONAL STANDARD

ISO/IEC 8824-1

First edition
2002-12-15

AMENDMENT 1
2004-10-01

Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation

AMENDMENT 1: Support for EXTENDED-XER

Technologies de l'information — Notation de syntaxe abstraite numéro un (ASN.1): Spécification de la notation de base

AMENDEMENT 1: Support pour règles de codage XML étendu

Reference number
ISO/IEC 8824-1:2002/Amd.1:2004(E)

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)	Introduction	1
2)	Subclause 2.2	1
3)	New subclause 3.6.18 *bis*	1
4)	New subclauses 3.6.22 *bis* to 3.6.22 *quat*	1
5)	Subclause 3.6.54	2
6)	Subclause 3.6.69	2
7)	Subclause 3.6.71	2
8)	New subclause 3.6.74 *bis*	2
9)	Subclase 8.1	2
10)	Subclause 8.3	2
11)	New subclause 8 *bis*	3
12)	New subclause 11.20 *bis*	3
13)	Subclause 11.23.2	3
14)	New subclause 11.23 *bis*	3
15)	Subclause 11.24.2	4
16)	New subclauses 11.24 *bis* to 11.24 *quat*	4
17)	Subclause 11.25.5	5
18)	Table 4	5
19)	Subclause 11.27	5
20)	Subclause 12.1	5
21)	Subclause 12.2	5
22)	New subclause 12.4 *bis*	5
23)	New subclause 12.21	6
24)	Subclause 13.3	6
25)	Subclause 16.2	6
26)	Subclause 16.9	6
27)	Subclause 16.10	6
28)	Subclause 17.3	6
29)	New subclause 17.4	6
30)	Subclause 18.9	7
31)	New subclause 18.9 *bis*	7
32)	Subclause 18.10	7
33)	New subclause 18.12	7
34)	Subclause 19.8	7
35)	New subclause 19.8 *bis*	8
36)	Subclause 19.9	8
37)	Subclauses 20.3 to 20.6	8
38)	New subclause 20.6 *bis*	9
39)	Subclause 20.7	9
40)	Subclause 21.9	10
41)	New subclause 21.9 *bis*	10
42)	Subclause 21.12	10
43)	New subclause 24.1 *bis*	10
44)	Subclause 24.2	11

		Page
45)	Subclause 24.3	11
46)	Subclause 24.5.1	11
47)	Subclause 24.8	11
48)	Subclause 24.9	11
49)	Subclause 25.3	11
51)	Table 5	11
52)	New subclauses 25.5 *bis* and 25.5 *ter*	12
53)	Subclauses 25.6, 25.7 and 25.8	12
54)	Subclause 25.11	12
55)	New subclauses 25.11.1 to 25.11.3	12
56)	Subclause 26.3	13
57)	Subclause 27.3	13
58)	Subclause 28.2	13
59)	Subclause 28.3	13
60)	Subclause 28.4	13
61)	Subclause 28.5	14
62)	Clause 30	14
63)	Clause 38	16
64)	New subclauses 38.1.4 *bis* and 38.1.4 *ter*	16
65)	Subclause 45.4	17
66)	Subclause 47.42	17
67)	Subclause 48.7.1	17
68)	New clause 50	17
69)	Subclause A.2.4	18
70)	Subclause B.3.2.2	18
71)	Subclause B.4.2	18
72)	New Annex C *bis*	18
73)	Subclause E.2.12.2	19
74)	Annex H	19

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 8824-1:2002 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 Amendment 1 to ITU-T Rec. X.680.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation

Amendment 1

Support for EXTENDED-XER

NOTE – All new or changed text in this amendment is underlined in the clauses being replaced. When merging all such text into the base Recommendation, the underlining is to be removed.

1) Introduction

a) *In the Introduction, insert the following paragraph immediately prior to the paragraph that begins with "An ASN.1 specification will initially be produced with a set of fully defined ASN.1 types.":*

It is also possible to assign encoding instructions to a type in order to affect the encoding of that type. This can be done either by a type prefix placed before a type definition or use of a type reference, or by an encoding control section placed at the end of an ASN.1 module. The generic syntax of type prefixes and encoding control sections is specified in this Recommendation | International Standard, and includes an encoding reference to identify the encoding rules that are modified by the encoding instruction. The semantics and detailed syntax of encoding instructions are specified in the encoding rules Recommendation | International Standard identified by the encoding reference.

b) *In the Introduction, insert the following paragraph between the paragraphs beginning with "Annex C" and "Annex D":*

Annex C *bis* forms an integral part of this Recommendation | International Standard and specifies the currently defined encoding references and the Recommendation | International Standard that defines the semantics and detailed syntax of encoding instructions with those encoding references.

2) Subclause 2.2

In subclause 2.2, replace the Note 1 with the following:

NOTE 1 – The above reference is included because it provides names for control characters and specifies categories of characters.

3) New subclause 3.6.18 *bis*

Insert a new subclause 3.6.18 bis as follows:

3.6.18 *bis* **default encoding reference (for a module)**: An encoding reference that is specified in the module header and is assumed in all type prefixes which do not contain an encoding reference.

NOTE – If a default encoding reference is not specified in the module header, then all type prefixes which do not contain an encoding reference are assigning tags.

4) New subclauses 3.6.22 *bis* to 3.6.22 *quat*

Insert three new subclauses 3.6.22 bis to 3.6.22 quat as follows:

3.6.22 *bis* **encoding control section**: Part of an ASN.1 module that enables encoding instructions to be assigned to types defined or used within that ASN.1 module.

3.6.22 *ter* **encoding instruction**: Information which can be associated with a type using a type prefix or an encoding control section, and which affects the encoding of that type by one or more ASN.1 encoding rules.

NOTE – An encoding instruction does not affect the abstract values of a type, and is not expected to be visible to an application.

3.6.22 *quat* **encoding reference**: A name (see Annex C *bis*) that identifies which encoding rules are affected by an encoding instruction in a type prefix or an encoding control section.

NOTE – The encoding reference **`TAG`** can be used to specify that a type prefix is assigning a tag rather than an encoding instruction (see 30.2).

5) Subclause 3.6.54

Replace subclause 3.6.54 with the following:

3.6.54 **real type**: A simple type whose distinguished values (specified in clause 20) include the set of real numbers (numerical real numbers) together with special values such as **`NOT-A-NUMBER`**.

6) Subclause 3.6.69

Replace subclause 3.6.69 with the following:

3.6.69 **tag**: Additional information, separate from the abstract values of the type, which is associated with every ASN.1 type and which can be changed or augmented by a type prefix.

NOTE – Tag information is used in some encoding rules to ensure that encodings are not ambiguous. Tag information differs from encoding instructions because tag information is associated with all ASN.1 types, even if they do not have a type prefix.

7) Subclause 3.6.71

Replace subclause 3.6.71 with the following:

3.6.71 **tagging**: Assigning a new tag to a type, replacing or adding to the existing (possibly the default) tag.

8) New subclause 3.6.74 *bis*

Insert a new subclause 3.6.74 bis as follows:

3.6.74 *bis* **type prefix**: Part of the ASN.1 notation that can be used to assign an encoding instruction or a tag to a type.

9) Subclause 8.1

Replace subclause 8.1 with the following:

8.1 A tag is specified (either within the text of this Recommendation | International Standard or by using a type prefix) by giving a class and a number within the class. The class is one of:

– universal;
– application;
– private;
– context-specific.

10) Subclause 8.3

*In subclause 8.3 replace "clause 30" with "*30.2*", and in the Note replace "*Clause 30*" with "*Subclause 30.2*".*

11) New subclause 8 *bis*

Insert a new clause 8 bis as follows:

8 *bis* Encoding instructions

8 *bis*.1 An encoding instruction is assigned to a type using either a type prefix (see 30.3) or an encoding control section (see clause 50).

8 *bis*.2 A type prefix may contain an encoding reference. If it does not, the encoding reference is determined by the default encoding reference for the module (see 12.4 *bis*).

8 *bis*.3 An encoding control section always contains an encoding reference. There may be multiple encoding control sections, but each encoding control section shall have a distinct encoding reference.

8 *bis*.4 An encoding instruction consists of a sequence of lexical items specified in the Recommendation | International Standard determined by the encoding reference (see Annex C *bis*).

8 *bis*.5 Multiple encoding instructions with the same or with different encoding references may be assigned to a type (using either or both of type prefixes and an encoding control section). Encoding instructions assigned with a given encoding reference are independent from those assigned with a different encoding reference, and from any use of a type prefix to perform tagging.

8 *bis*.6 The effect of assigning several encoding instructions with the same encoding reference (using either or both of type prefixes and an encoding control section) is specified in the Recommendation | International Standard determined by the encoding reference (see Annex C *bis*), and is not specified in this Recommendation | International Standard.

8 *bis*.7 If an encoding instruction is assigned to the "Type" in a "TypeAssignment", it becomes associated with the type, and is applied wherever the "typereference" of the "TypeAssignment" is used. This includes use in other modules through the export and import statements.

12) New subclause 11.20 *bis*

Insert a new subclause 11.20 bis as follows:

11.20 *bis* Encoding references

Name of item – encodingreference

An "encodingreference" shall consist of a sequence of characters as specified for a "typereference" in 11.2, except that no lower-case letters shall be included.

NOTE – Currently defined encoding references are listed in Annex C *bis* with the Recommendation | International Standard that specifies the syntax and semantics of the corresponding encoding instructions. The "encodingreference" shall consist only of the sequences listed in Annex C *bis* in this or in future versions of this Recommendation | International Standard.

13) Subclause 11.23.2

Replace subclause 11.23.2 with the following:

11.23.2 In analyzing an instance of use of this notation, a "true" is distinguished from a "valuereference" or an "identifier" or an instance of XML boolean "extended-true" by the context in which it appears.

NOTE – This sequence does not contain any white-space characters (see 11.1.2).

14) New subclause 11.23 *bis*

Insert a new subclause 11.23 bis as follows:

11.23 *bis* XML boolean extended-true item

Name of item – extended-true

11.23 *bis*.1 This item shall consist of either the sequence of characters:

true

or of the single character:

1 (DIGIT ONE)

11.23 *bis*.2 In analyzing an instance of use of this notation, an "extended-true" is distinguished from a "valuereference" or an "identifier" or an instance of XML boolean "true" by the context in which it appears.

NOTE – This sequence does not contain any white-space characters (see 11.1.2).

15) Subclause 11.24.2

Replace subclause 11.24.2 with the following:

11.24.2 In analyzing an instance of use of this notation, a "false" is distinguished from a "valuereference" or an "identifier" or an instance of XML boolean "extended-false" by the context in which it appears.

NOTE – This sequence does not contain any white-space characters (see 11.1.2).

16) New subclauses 11.24 *bis* to 11.24 *quat*

Insert new subclauses 11.24 bis, ter and quat as follows:

11.24 *bis* XML boolean extended-false item

Name of item – extended-false

11.24 *bis*.1 This item shall consist of either the sequence of characters:

false

or of the single character:

0 (DIGIT ZERO)

11.24 *bis*.2 In analyzing an instance of use of this notation, a "false" is distinguished from a "valuereference" or an "identifier" or an instance of XML boolean "false" by the context in which it appears.

NOTE – This sequence does not contain any white-space characters (see 11.1.2).

11.24 *ter* XML real not-a-number item

Name of item – "NaN"

11.24 *ter*.1 This item shall consist of the sequence of characters:

NaN

11.24 *ter*.2 In analyzing an instance of use of this notation, a "NaN" is distinguished from any other lexical item commencing with an upper-case letter by the context in which it appears.

NOTE – This sequence does not contain any white-space characters (see 11.1.2).

11.24 *quat* XML real infinity item

Name of item – "INF"

11.24 *quat*.1 This item shall consist of the sequence of characters:

INF

11.24 *quat*.2 In analyzing an instance of use of this notation, an "INF" is distinguished from any other lexical item commencing with an upper-case letter by the context in which it appears.

NOTE – This sequence does not contain any white-space characters (see 11.1.2).

17) Subclause 11.25.5

a) Replace subclause 11.25.5 with the following:

11.25.5 If the ASN.1 built-in type is a "PrefixedType", then the type which determines the "xmlasn1typename" shall be "Type" in the "PrefixedType" (see 30.1.5). If this is itself a "PrefixedType", then this subclause 11.25.5 shall be recursively applied.

NOTE – The subclauses of 25.11 specify the "Type" to be used for a "SelectionType" and a "ConstrainedType".

18) Table 4

*In Table 4, replace "*TaggedType*" with "*PrefixedType*".*

19) Subclause 11.27

In subclause 11.27, insert the following 3 new reserved words in new cells in the table in the appropriate alphabetical position:

ENCODING-CONTROL

INSTRUCTIONS

NOT-A-NUMBER

20) Subclause 12.1

In subclause 12.1, replace the production "ModuleDefinition" with the following:

```
ModuleDefinition ::=
        ModuleIdentifier
        DEFINITIONS
        EncodingReferenceDefault
        TagDefault
        ExtensionDefault
        "::="
        BEGIN
        ModuleBody
        EncodingControlSections
        END
```

and insert the following new production immediately before the "TagDefault" production:

```
EncodingReferenceDefault ::=
        encodingreference INSTRUCTIONS
    |   empty
```

21) Subclause 12.2

*In subclause 12.2 in the Note, replace "*Clause 30*" with "*Subclause 30.2*".*

22) New subclause 12.4 *bis*

Insert a new subclause 12.4 bis as follows:

12.4 ***bis*** The "EncodingReferenceDefault" specifies that the "encodingreference" is the default encoding reference for the module. If the "EncodingReferenceDefault" is "empty", then the default encoding reference for the module is **TAG**.

NOTE – Annex C *bis* contains a list of allowed encoding references, together with the Recommendation | International Standard which specifies the form and meaning of the corresponding encoding instructions.

23) New subclause 12.21

Insert a new subclause 12.21 as follows:

12.21 "EncodingControlSections" is specified in clause 50.

24) Subclause 13.3

*In subclause 13.3 in the Note, replace "*XML tags*" with "*XML tag names*".*

25) Subclause 16.2

*In subclause 16.2, replace the 2 occurrences of "*TaggedType*" with "*PrefixedType*".*

26) Subclause 16.9

*In subclause 16.9, replace "*TaggedValue*" with "*PrefixedValue*".*

27) Subclause 16.10

*In subclause 16.10, replace "*XMLTaggedValue*" with "*XMLPrefixedValue*".*

28) Subclause 17.3

Replace subclause 17.3 with the following:

17.3 The value of a boolean type (see 3.6.73 and 3.6.38) shall be defined by the notation "BooleanValue", or when used as an "XMLValue", by the notation "XMLBooleanValue". These productions are:

```
BooleanValue ::= TRUE | FALSE

XMLBooleanValue ::=
          EmptyElementBoolean
    |     TextBoolean

EmptyElementBoolean ::=
          "<" & "true" "/>"
    |     "<" & "false" "/>"

TextBoolean ::=
          extended-true
    |     extended-false
```

29) New subclause 17.4

Insert a new subclause 17.4 as follows:

17.4 If an "EmptyElementBoolean" appears in an "XMLValueAssignment", then there shall be no occurrence of "TextBoolean" in that "XMLValueAssignment".

30) Subclause 18.9

Replace subclause 18.9 with the following:

18.9 The value of an integer type shall be defined by the notation "IntegerValue", or when used as an "XMLValue", by the notation "XMLIntegerValue". These productions are:

IntegerValue ::=
SignedNumber |
identifier

XMLIntegerValue ::=
XMLSignedNumber
| EmptyElementInteger
| TextInteger

XMLSignedNumber ::=
number
| "-" & number

EmptyElementInteger ::=
"<" & identifier "/>"

TextInteger ::=
identifier

31) New subclause 18.9 *bis*

Add a new subclause 18.9 bis as follows:

18.9 *bis* If an "EmptyElementInteger" appears in an "XMLValueAssignment", then there shall be no occurrence of "TextInteger" in that "XMLValueAssignment".

32) Subclause 18.10

Replace subclause 18.10 and its Note with the following:

18.10 The "identifier" in "IntegerValue" and in the last two alternatives for "XMLIntegerValue" shall be one of the "identifier"s in the "IntegerType" with which the value is associated, and shall represent the corresponding number.

NOTE – When referencing an integer value for which an "identifier" has been defined, use of the "identifier" form of "IntegerValue" and one of the "identifier" forms of "XMLIntegerValue" should be preferred.

33) New subclause 18.12

Add a new subclause 18.12 as follows:

18.12 The second alternative of "XMLSignedNumber" shall not be used if the "number" is zero.

34) Subclause 19.8

Replace subclause 19.8 with the following:

19.8 The value of an enumerated type shall be defined by the notation "EnumeratedValue", or when used as an "XMLValue", by the notation "XMLEnumeratedValue". These productions are:

EnumeratedValue ::= identifier

XMLEnumeratedValue ::=
EmptyElementEnumerated
| TextEnumerated

EmptyElementEnumerated ::= "<" & identifier "/>"

TextEnumerated ::= identifier

35) New subclause 19.8 *bis*

Add a new subclause 19.8 bis as follows:

19.8 *bis* If an "EmptyElementEnumerated" appears in an "XMLValueAssignment", then there shall be no occurrence of "TextEnumerated" in that "XMLValueAssignment".

36) Subclause 19.9

Replace subclause 19.9 with the following:

19.9 The "identifier" in "EnumeratedValue" and in the two alternatives of "XMLEnumeratedValue" shall be equal to that of an "identifier" in the "EnumeratedType" sequence with which the value is associated.

37) Subclauses 20.3 to 20.6

Replace subclauses 20.3, 20.4, 20.5 and 20.6 with the following:

20.3 The abstract values of the real type are the special values **PLUS-INFINITY**, **MINUS-INFINITY**, and **NOT-A-NUMBER** together with numeric real numbers consisting of either plus zero or minus zero, or capable of being specified by the following formula involving three integers, M, B and E:

$$M \times B^E$$

where M (non-zero) is called the mantissa, B (either 2 or 10) the base, and E the exponent. Values with B = 2 ("**base**" 2 abstract values) and B = 10 ("**base**" 10 abstract values) are defined as distinct abstract values. Otherwise, values of $M \times B^E$ which evaluate to the same numerical value are a single abstract value.

NOTE – Minus zero and plus zero are two distinct abstract values for a mathematical zero, and the "**base**" 2 and "**base**" 10 abstract values are distinct abstract values for all other numeric real numbers.

20.4 The real type has an associated type which is used to support the value and subtype notations for numeric values of the real type (in addition to the notation for the special values of the real type and for plus zero and minus zero).

NOTE – Encoding rules may define a different type which is used to specify encodings, or may specify encodings without reference to the associated type. In particular, the encoding in BER and PER provides a Binary-Coded Decimal (BCD) encoding if "base" is 10, and an encoding which permits efficient transformation to and from hardware floating point representations if "base" is 2.

20.5 The associated type for value definition (and for subtyping purposes) of the numeric values is (with normative comments):

```
SEQUENCE {
        mantissa    INTEGER (ALL EXCEPT 0),
        base  INTEGER (2|10),
        exponent    INTEGER

        -- The associated mathematical real number is "mantissa"

        -- multiplied by "base" raised to the power "exponent"
}
```

NOTE 1 – Values represented by "**base**" 2 and by "**base**" 10 are considered to be distinct abstract values even if they evaluate to the same real number value, and may carry different application semantics.

NOTE 2 – The notation **REAL (WITH COMPONENTS { ... , base (10)})** can be used to restrict the set of values to the "**base**" 10 numeric values (and similarly for "**base**" 2 numeric values). This notation does not include the values (special real values and plus and minus zero) that cannot be represented by the associated type. If required, these can be added using set arithmetic.

NOTE 3 – This type is capable of carrying an exact finite representation of any number which can be stored in typical floating point hardware, and of any number with a finite character-decimal representation.

20.6 The value of a real type shall be defined by the notation "RealValue", or when used in an "XMLValue", by the notation "XMLRealValue":

RealValue ::=
    NumericRealValue
  | **SpecialRealValue**

NumericRealValue ::=
    realnumber
  | **"-" realnumber**
  | **SequenceValue** *-- Value of the associated sequence type*

SpecialRealValue ::=
    PLUS-INFINITY
  | **MINUS-INFINITY**
  | **NOT-A-NUMBER**

NOTE – The third alternative of "NumericRealValue" cannot be used for plus zero or minus zero values. These abstract values are specified using either the first or the second alternative respectively, with a single "**0**" character for "realnumber".

XMLRealValue ::=
    XMLNumericRealValue | XMLSpecialRealValue

XMLNumericRealValue ::=
    realnumber
  | **"-" & realnumber**

XMLSpecialRealValue ::=
    EmptyElementReal
  | **TextReal**

EmptyElementReal ::=
    "<" & PLUS-INFINITY "/>"
  | **"<" & MINUS-INFINITY "/>"**
  | **"<" & NOT-A-NUMBER "/>"**

TextReal ::=
    "INF"
  | **"-" & "INF"**
  | **"NaN"**

38) New subclause 20.6 *bis*

Add a new subclause 20.6 bis as follows:

20.6 *bis* If an "EmptyElementReal" appears in an "XMLValueAssignment", then there shall be no occurrence of "TextReal" in that "XMLValueAssignment".

39) Subclause 20.7

Replace subclause 20.7 with the following:

20.7 When the "realnumber" notation is used, it identifies the corresponding "**base**" 10 abstract value, or plus zero. When a "realnumber" value is preceded by "**-**", it identifies the corresponding "**base**" 10 abstract values that are negative numbers, or minus zero. If the "RealType" is constrained to "**base**" 2, the "realnumber" or "**-**" "realnumber" identifies the "**base**" 2 abstract value corresponding either to the decimal value specified by the "realnumber" or to a locally-defined precision if an exact representation is not possible.

40) Subclause 21.9

Replace subclause 21.9 with the following:

21.9 The value of a bitstring type shall be defined by the notation "BitStringValue", or when used as an "XMLValue", by the notation "XMLBitStringValue". These productions are:

BitStringValue ::=
bstring
| **hstring**
| **"{" IdentifierList "}"**
| **"{" "}"**
| **CONTAINING Value**

IdentifierList ::=
identifier
| **IdentifierList "," identifier**

XMLBitStringValue ::=
XMLTypedValue
| **xmlbstring**
| **XMLIdentifierList**
| **empty**

XMLIdentifierList ::=
EmptyElementList
| **TextList**

EmptyElementList ::=
"<" & identifier "/>"
| **EmptyElementList "<" & identifier "/>"**

TextList ::
identifier
| **TextList identifier**

41) New subclause 21.9 *bis*

Add a new subclause 21.9 bis as follows:

21.9 *bis* If an "EmptyElementList" appears in an "XMLValueAssignment", then there shall be no occurrence of "TextList" in that "XMLValueAssignment".

42) Subclause 21.12

Replace subclause 21.12 with the following:

21.12 Each "identifier" in "BitStringValue" or in the alternatives of "XMLBitStringValue" shall be the same as an "identifier" in the "BitStringType" production sequence with which the value is associated.

43) New subclause 24.1 *bis*

Add a new subclause 24.1 bis as follows:

24.1 *bis* For the purposes of the following clauses, a "PrefixedType" is defined to be a textually tagged type if either:

a) the "PrefixedType" is a "TaggedType"; or

b) the "Type" in the "PrefixedType" is a textually tagged type.

44) Subclause 24.2

Replace the text of subclause 24.2 with the following (retaining the two Notes):

24.2 When the "ComponentTypeLists" production occurs within the definition of a module for which automatic tagging is selected (see 12.3), and none of the occurrences of "NamedType" in any of the first three alternatives for "ComponentType" is a textually tagged type (see 24.1 *bis*), then the automatic tagging transformation is selected for the entire "ComponentTypeLists"; otherwise, it is not.

45) Subclause 24.3

*In the Note of subclause 24.3, replace "*explicitly*" with "*textually*".*

46) Subclause 24.5.1

*In subclause 24.5.1, replace "*clause 30*" with "*30.2*".*

47) Subclause 24.8

Replace subclause 24.8 with the following:

24.8 If automatic tagging is in effect and the "ComponentType"s in the extension root have no tags, then no "ComponentType" within the "ExtensionAdditionList" shall be a textually tagged type.

48) Subclause 24.9

*In Note 1 of subclause 24.9, replace "*30.6*" with "*30.2.7*".*

49) Subclause 25.3

In subclause 25.3, remove the "XMLSpaceSeparatedList" alternative from the "XMLSequenceOfValue" production, remove the "XMLSpaceSeparatedList" production, remove "1" in Note 1, and delete Note 2.

50) Subclause 25.4

Add the following Note after subclause 25.4:

NOTE – This occurs only for **`SEQUENCE OF NULL`**.

51) Table 5

Replace Table 5 with the following:

Table 5 – "XMLSequenceOfValue" and "XMLSetOfValue" notation for ASN.1 types

ASN.1 type	XML value notation
BitStringType	XMLDelimitedItemList
BooleanType	*See 25.5 bis.*
CharacterStringType	XMLDelimitedItemList
ChoiceType	XMLValueList
EmbeddedPDVType	XMLDelimitedItemList
EnumeratedType	*See 25.5 ter.*
ExternalType	XMLDelimitedItemList
InstanceOfType	*See ITU-T Rec. X.681 | ISO/IEC 8824-2, C.9.*

ASN.1 type	XML value notation
IntegerType	XMLDelimitedItemList
NullType	XMLValueList
ObjectClassFieldType	*See ITU-T Rec. X.681 | ISO/IEC 8824-2, 14.10 and 14.11.*
ObjectIdentifierType	XMLDelimitedItemList
OctetStringType	XMLDelimitedItemList
RealType	XMLDelimitedItemList
RelativeOIDType	XMLDelimitedItemList
SequenceType	XMLDelimitedItemList
SequenceOfType	XMLDelimitedItemList
SetType	XMLDelimitedItemList
SetOfType	XMLDelimitedItemList
PrefixedType	*See 25.11.1.*
UsefulType (GeneralizedTime)	XMLDelimitedItemList
UsefulType (UTCTime)	XMLDelimitedItemList
UsefulType (ObjectDescriptor)	XMLDelimitedItemList
TypeFromObject	*See ITU-T Rec. X.681 | ISO/IEC 8824-2, 15.6.*
ValueSetFromObjects	*See ITU-T Rec. X.681 | ISO/IEC 8824-2, 15.6.*

52) New subclauses 25.5 *bis* and 25.5 *ter*

Insert two new subclauses 25.5 bis and 25.5 ter as follows:

25.5 *bis* If "EmptyElementBoolean" is used for the value of a boolean type, then "XMLValueList" shall be used; otherwise, "XMLDelimitedItemList" shall be used.

25.5 *ter* If "EmptyElementEnumerated" is used for the value of an enumerated type, then "XMLValueList" shall be used; otherwise, "XMLDelimitedItemList" shall be used.

53) Subclauses 25.6, 25.7 and 25.8

Delete the three subclauses 25.6, 25.7 and 25.8. Note that these clauses have been inserted with minor modifications as 25.11.1, 25.11.2 and 25.11.3 respectively.

54) Subclause 25.11

Replace subclause 25.11 with the following:

25.11 If the first alternative of "XMLDelimitedItem" is used, then if the component of the sequence-of type (after ignoring any occurrences of "TypePrefix") is a "typereference" or an "ExternalTypeReference", then the "NonParameterizedTypeName" shall be the "typereference" or the "typereference" in the "ExternalTypeReference", respectively; otherwise, it shall be the "xmlasn1typename" specified in Table 4 corresponding to the built-in type of the component.

55) New subclauses 25.11.1 to 25.11.3

Insert three new subclauses 25.11.1, 25.11.2 and 25.11.3 as follows:

25.11.1 If the "Type" of the component is a "PrefixedType", then the type which determines the "XMLSequenceOfValue" alternative and the "xmlasn1typename" (if required) shall be the "Type" in the "PrefixedType" (see 30.1.5). If this is itself a "PrefixedType", a "ConstrainedType" or a "SelectionType", then these subclauses of 25.11 shall be recursively applied.

25.11.2 If the "Type" of the component is a "ConstrainedType", then the type which determines the "XMLSequenceOfValue" alternative and the "xmlasn1typename" (if required) shall be the "Type" in the "ConstrainedType" (see 45.1). If this is itself a "PrefixedType", a "ConstrainedType" or a "SelectionType", then these subclauses of 25.11 shall be recursively applied.

25.11.3 If the "Type" of the component is a "SelectionType", then the type which determines the "XMLSequenceOfValue" alternative and the "xmlasn1typename" (if required) notation shall be the type referenced by the "SelectionType" (see clause 29). If this is itself a "PrefixedType", a "ConstrainedType" or a "SelectionType", then these subclauses of 25.11 shall be recursively applied.

56) Subclause 26.3

*In subclause 26.3, replace "*clause 30*" with "*30.2*".*

57) Subclause 27.3

Replace subclause 27.3 with the following:

27.3 The notation for defining a value of a set-of type shall be the "SetOfValue", or when used as an "XMLValue", "XMLSetOfValue". These productions are:

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

XMLSetOfValue ::=
XMLValueList
| **XMLDelimitedItemList**
| **empty**

"ValueList", "NamedValueList" and the alternatives of "XMLSetOfValue" are specified in 25.3, and the choice of alternative is the same as if "XMLSequenceOfValue" had been used. The **"{" "}"** or "empty" notation is used when the "SetOfValue" or "XMLSetOfValue" is an empty list.

NOTE 1 – Semantic significance should not be placed on the order of these values.

NOTE 2 – Encoding rules are not required to preserve the order of these values.

NOTE 3 – The set-of type is not a mathematical set of values, thus, as an example, for **SET OF INTEGER** the values **{ 1 }** and **{ 1 1 }** are distinct.

58) Subclause 28.2

Replace subclause 28.2 with the following:

28.2 When the "AlternativeTypeLists" production occurs within the definition of a module for which automatic tagging is selected (see 12.3), and none of the occurrences of "NamedType" in any "AlternativeTypeList" is a textually tagged type (see 24.1 *bis*), the automatic tagging transformation is selected for the entire "AlternativeTypeLists"; otherwise, it is not.

59) Subclause 28.3

*In subclause 28.3 replace "*clause 30*" with "*30.2*".*

60) Subclause 28.4

*In subclause 28.4, replace "*tagged type*" with "*textually tagged type*".*

61) Subclause 28.5

*In subclause 28.5, Note 1, replace "*30.6*" with "*30.2.7*".*

62) Clause 30

Replace clause 30 and its subclauses with the following:

30 Notation for prefixed types

30.1 General

30.1.1 A prefixed type is a new type which is isomorphic with an old type, but which has a different or additional tag and may have different or additional associated encoding instructions.

30.1.2 A prefixed type is either a "TaggedType" or an "EncodingPrefixedType".

30.1.3 A prefixed type which is a tagged type is mainly of use where this Recommendation | International Standard requires the use of types with distinct tags (see 24.5 to 24.6, 26.3 and 28.3). The use of a "TagDefault" of **AUTOMATIC TAGS** in a module allows this to be accomplished without the explicit appearance of "TaggedType" in that module.

NOTE – Where a protocol determines that values from several data types may be transmitted at any moment in time, distinct tags may be needed to enable the recipient to correctly decode the value.

30.1.4 The assignment of an encoding instruction using an "EncodingPrefixedType" is only relevant to the encodings identified by the associated encoding reference and has no effect on the abstract values of the type.

30.1.5 The notation for a prefixed type shall be "PrefixedType":

```
PrefixedType ::=
        TaggedType
|       EncodingPrefixedType
```

NOTE – Specification of the syntax for "PrefixedType" would be simpler and clearer if tagging was described as the assignment of an encoding instruction. However, historically, tagging was introduced in the earliest versions of the ASN.1 specifications, and can affect the legality of a type definition. Minimum changes to the concepts of tagging (and the associated syntactic descriptions) were made when encoding prefixed types were introduced. Tagging also differs syntactically from assignment of encoding instructions: the specification that tagging is **EXPLICIT** or **IMPLICIT** occurs following the closing "**]**" of the tag, it is not contained within the paired "**[**" and "**]**" as is the case with normal encoding instructions.

30.1.6 The notation for a value of a "PrefixedType" shall be "PrefixedValue", or when used as an "XMLValue", "XMLPrefixedValue". These productions are:

```
PrefixedValue ::= Value

XMLPrefixedValue ::= XMLValue
```

where "Value" or "XMLValue" is a notation for a value of the "Type" in the "TaggedType" or the "EncodingPrefixedType" of the "PrefixedType".

NOTE 1 – Neither the "Tag" nor any part of the "EncodingPrefix" appears in this notation.

NOTE 2 – Encoding instructions can also be assigned to a type in an encoding control section (see clause 50). Such an assignment has no effect on the value notation for a type.

30.2 The tagged type

30.2.1 The notation for a tagged type shall be "TaggedType":

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

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

EncodingReference ::=
        encodingreference ":"
|       empty
```

ClassNumber ::=
    number
  | **DefinedValue**

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

30.2.2 When used in "Tag", the "encodingreference" shall be **TAG**. The "EncodingReference" in "Tag" shall not be "empty" unless the default encoding reference for the module is **TAG** (see 12.4 *bis*).

30.2.3 The "valuereference" in "DefinedValue" shall be of type integer, and assigned a non-negative value.

30.2.4 The new type is isomorphic with the old type, but has a tag with class "Class" and number "ClassNumber", except when "Class" is "empty", in which case the tag is context-specific class and number is "ClassNumber".

30.2.5 The "Class" shall not be **UNIVERSAL** except for types defined in this Recommendation | International Standard.

NOTE 1 – Use of universal class tags are agreed from time to time by ITU-T and ISO.

NOTE 2 – Subclause E.2.12 contains guidance and hints on stylistic use of tag classes.

30.2.6 All application of tags is either implicit tagging or explicit tagging. Implicit tagging indicates, for those encoding rules which provide the option, that explicit identification of the original tag of the "Type" in the "TaggedType" is not needed during transfer.

NOTE – It can be useful to retain the old tag where this was universal class, and hence unambiguously identifies the old type without knowledge of the ASN.1 definition of the new type. Minimum transfer octets is, however, normally achieved by the use of **IMPLICIT**. An example of an encoding using **IMPLICIT** is given in ITU-T Rec. X.690 | ISO/IEC 8825-1.

30.2.7 The tagging construction specifies explicit tagging if any of the following holds:

a) the "Tag **EXPLICIT** Type" alternative is used;

b) the "Tag Type" alternative is used and the value of "TagDefault" for the module is either **EXPLICIT TAGS** or is empty;

c) the "Tag Type" alternative is used and the value of "TagDefault" for the module is **IMPLICIT TAGS** or **AUTOMATIC TAGS**, but the type defined by "Type" is an untagged choice type, an untagged open type, or an untagged "DummyReference" (see ITU-T Rec. X.683 | ISO/IEC 8824-4, 8.3).

The tagging construction specifies implicit tagging otherwise.

30.2.8 If the "Class" is "empty", there are no restrictions on the use of "Tag", other than those implied by the requirement for distinct tags in 24.5 to 24.6, 26.3 and 28.3.

30.2.9 The **IMPLICIT** alternative shall not be used if the type defined by "Type" is an untagged choice type or an untagged open type or an untagged "DummyReference" (see ITU-T Rec. X.683 | ISO/IEC 8824-4, 8.3).

30.3 The encoding prefixed type

30.3.1 The notation for an encoding prefixed type shall be "EncodingPrefixedType":

EncodingPrefixedType ::=
    EncodingPrefix Type

EncodingPrefix ::=
    "[" EncodingReference EncodingInstruction "]"

"EncodingReference" is defined in 30.2.1.

30.3.2 The "EncodingInstruction" production is specified in the Recommendation | International Standard identified by the "EncodingReference" (see Annex C *bis*) and can consist of any sequence of ASN.1 lexical items (including comment, cstring and white-space).

NOTE 1 – The "**[**" and "**]**" lexical items never appear in "EncodingInstruction".

NOTE 2 – Future versions of this Recommendation | International Standard may add further encoding references to Annex C *bis*. It is recommended that ASN.1 tools provide (only) warnings if an "encodingreference" is not one of those specified in Annex C *bis* and then ignore the whole "EncodingPrefix" using a "**]**" as the terminator (see Note 1 above).

30.3.3 If the "EncodingReference" is empty, then the encoding reference for the encoding prefix is the default encoding reference for the module.

NOTE – If the default encoding reference for the module is **TAG** (see 30.2.2) and the "EncodingReference" is "empty", then the "PrefixedType" is a "TaggedType", not an "EncodingPrefixedType".

30.3.4 There are in general restrictions on the encoding instructions (with the same encoding reference) that can be used in combination, and on the types to which particular instructions or combinations of instructions can be applied. These restrictions are specified in the Recommendation | International Standard associated with the encoding reference (see Annex C *bis*), and are not specified in this Recommendation | International Standard.

63) Clause 38

Replace clause 38 with the following:

38 Naming characters, collections and property category sets

This clause specifies an ASN.1 built-in module which contains the definition of a value reference name for each character from ISO/IEC 10646-1, where each name references a **UniversalString** value of size 1. This module also contains the definition of a type reference name for each collection of characters from ISO/IEC 10646-1, where each name references a subset of the **UniversalString** type. Finally, it contains the definition of a "typereference" name for the set of characters in each general category of character properties that are listed in 4.5 of The Unicode Standard, where each name references a subset of the **UniversalString** type.

NOTE – These values are available for use in the value notation of the **UniversalString** type and types derived from it. All of the value and type references defined in the module specified in 38.1 are exported and must be imported by any module that uses them.

64) New subclauses 38.1.4 *bis* and 38.1.4 *ter*

Insert two new subclauses 38.1.4 bis and 38.1.4 ter as follows:

38.1.4 *bis* For each abbreviation and each description listed in The Unicode Standard, Table 4-5, two statements are included in the module of the form:

```
<categoryabbreviation> ::= UniversalString (FROM (<alternativelist>))
      -- represents the set of characters with the property
      -- category <categoryabbreviation>.

<categorydescription> UniversalString ::= <categoryabbreviation>
```

where:

a) <categoryabbreviation> is the abbreviation for the general category of character properties listed in The Unicode Standard, Table 4-5 (for example, **Lu** or **Nd** or **Pi**);

b) <categorydescription> is the description for the same general category of characters, with the initial letter of all words uppercased, the comma and all spaces removed, and all description in parentheses removed (for example, **LetterUppercase** or **NumericDigit** or **PunctuationInitialQuote**);

c) The <alternativelist> for each <categoryabbreviation> is a list of the <namedcharacter> names produced by 38.2 for each of the characters listed in The Unicode Character Database (version 3.2.0) of The Unicode Standard that have the corresponding <categoryabbreviation>.

NOTE – The Unicode name for a character is the same as the <iso10646name> for that character.