INTERNATIONAL TELECOMMUNICATION UNION

ITU-T X.693

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Amendment 1
(10/2003)

SERIES X: DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

OSI networking and system aspects – Abstract Syntax Notation One (ASN.1)

Information technology – ASN.1 encoding rules: XML Encoding Rules (XER)

Amendment 1: XER encoding instructions and EXTENDED-XER

ITU-T Recommendation X.693 (2001) – Amendment 1

ITU-T X-SERIES RECOMMENDATIONS

DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

PUBLIC DATA NETWORKS	
Services and facilities	X.1–X.19
Interfaces	X.20–X.49
Transmission, signalling and switching	X.50–X.89
Network aspects	X.90–X.149
Maintenance	X.150–X.179
Administrative arrangements	X.180–X.199
OPEN SYSTEMS INTERCONNECTION	
Model and notation	X.200–X.209
Service definitions	X.210–X.219
Connection-mode protocol specifications	X.220–X.229
Connectionless-mode protocol specifications	X.230–X.239
PICS proformas	X.240–X.259
Protocol Identification	X.260–X.269
Security Protocols	X.270–X.279
Layer Managed Objects	X.280–X.289
Conformance testing	X.290–X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300–X.349
Satellite data transmission systems	X.350–X.369
IP-based networks	X.370–X.399
MESSAGE HANDLING SYSTEMS	X.400–X.499
DIRECTORY	X.500–X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600–X.629
Efficiency	X.630–X.639
Quality of service	X.640–X.649
Naming, Addressing and Registration	X.650–X.679
Abstract Syntax Notation One (ASN.1)	**X.680–X.699**
OSI MANAGEMENT	
Systems Management framework and architecture	X.700–X.709
Management Communication Service and Protocol	X.710–X.719
Structure of Management Information	X.720–X.729
Management functions and ODMA functions	X.730–X.799
SECURITY	X.800–X.849
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850–X.859
Transaction processing	X.860–X.879
Remote operations	X.880–X.899
OPEN DISTRIBUTED PROCESSING	X.900–X.999

For further details, please refer to the list of ITU-T Recommendations.

INTERNATIONAL STANDARD ISO/IEC 8825-4
ITU-T RECOMMENDATION X.693

Information technology – ASN.1 encoding rules: XML Encoding Rules (XER)

Amendment 1

XER encoding instructions and EXTENDED-XER

Summary

An Amendment 1 is provided for ITU-T Rec. X.680 | ISO/IEC 8824-1, ITU-T Rec. X.681 | ISO/IEC 8824-2, ITU-T Rec. 690 | ISO/IEC 8825-1, ITU-T Rec. X.691 | ISO/IEC 8825-2 and ITU-T Rec. X.693 | ISO/IEC 8825-4. These amendments provide the following:

- Correction of a bug in CXER resulting from allowing white-space between a minus sign and a following **`INTEGER`** or **`REAL`** value (CXER was not canonical). This is no longer permitted, in value notation, XML Value Notation or in XER and CXER. **This is a change** (introduced by text in Amendment 1 to Rec. X.680 | ISO/IEC 8824-1) **and not an addition**.
- Addition of encoding instructions in an ASN.1 module, using either a type prefix or within an encoding control section, in order to specify variations of the BASIC-XER encodings. These encoding instructions are designed to support mappings from an XSD specification to an ASN.1 specification. This provision has meant a change of terminology, where a type with "**`[`**…**`]`**" in front of it is a prefixed type, and the "**`[`**…**`]`**" notation may or may not be a tag. This change of terminology results in changes to the text (but not the substance) of the BER and PER specifications, so there is also an Amendment 1 to these specifications.
- The addition of NaN (Not-a-Number) and minus zero as new values for **`REAL`** (support for encoding these new values is provided in Amendment 1 to ITU-T Rec. X.690 | ISO/IEC 8825-1 and to ITU-T Rec. X.691 | ISO/IEC 8825-2, as well as in Amendment 1 to ITU-T Rec. X.693 | ISO/IEC 8825-4).
- The addition of new XML Value Notations for **`REAL`**, **`BOOLEAN`**, **`ENUMERATED`**, and **`INTEGER`** that use text rather than empty-element tags for the values. These are available in XML Value Notation and in EXTENDED-XER, but not in BASIC-XER (for reasons of backwards-compatibility).
- Changes to the XML Value Notation for sequence-of (and the XER encodings) to provide delimitation of values where they are not XML elements (this occurs with the additional XML Value Notations, and only affects use of those additional XML Value Notations). This change is only concerned with use of XML Value Notations that have been added by this amendment, and these are not allowed in BASIC-XER, which is not affected.

This provides the necessary basic support for EXTENDED-XER.

This amendment depends on ITU-T Rec. X.680 (2002)/Amd.1 (2003) | ISO/IEC 8824-1:2002/Amd.1:2003 and on ITU-T Rec. X.681 (2002)/Amd.1 (2003) | ISO/IEC 8824-2:2002/Amd.1:2003 for the provision of alternative encodings for some types and for syntax for the insertion of XER encoding instructions in an ASN.1 specification. Many references from this amendment to clauses in these Recommendations | International Standards are to clauses introduced by those amendments.

The bulk of this amendment is concerned with the specification of the syntax and semantics of (new) XER encoding instructions that can be used to require EXTENDED-XER encoders to provide specialized encodings for ASN.1 types. These specialized encodings are largely designed to support ITU-T Rec. X.694 | ISO/IEC 8825-5 (Mapping XSD into ASN.1).

Source

Amendment 1 to ITU-T Recommendation X.693 (2001) was approved on 29 October 2003 by ITU-T Study Group 17 (2001-2004) under the ITU-T Recommendation A.8 procedure. An identical text is also published as ISO/IEC 8825-4, Amendment 1.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – ASN.1 encoding rules: XML Encoding Rules (XER)

Amendment 1

XER encoding instructions and EXTENDED-XER

NOTE – All new or changed text in this amendment is underlined in clauses being replaced (but not in new clauses). When merging all such text into the base Recommendation, the underlining is to be removed.

Replace the Introduction with the following:

Introduction

The publications ITU-T Rec. X.680 | ISO/IEC 8824-1, ITU-T Rec. X.681 | ISO/IEC 8824-2, ITU-T Rec. X.682 | ISO/IEC 8824-3, ITU-T Rec. X.683 | ISO/IEC 8824-4 together describe Abstract Syntax Notation One (ASN.1), a notation for the definition of messages to be exchanged between peer applications.

This Recommendation | International Standard defines encoding rules that may be applied to values of ASN.1 types defined using the notation specified in the Recommendations | International Standards listed above. Application of these encoding rules produces a transfer syntax for such values. It is implicit in the specification of these encoding rules that they are also to be used for decoding.

There is more than one set of encoding rules that can be applied to values of ASN.1 types. This Recommendation | International Standard defines three sets of encoding rules that use the Extensible Markup Language (XML). These encoding rules all produce an XML document compliant to W3C XML 1.0. The first set is called the basic XML Encoding Rules (BASIC-XER). The second set is called the Canonical XML Encoding Rules (CANONICAL-XER, or CXER) because there is only one way of encoding an ASN.1 value using these encoding rules. (Canonical encoding rules are generally used for applications using security-related features such as digital signatures.) The third set is called the extended XML Encoding Rules (EXTENDED-XER). The extended XML Encoding Rules allow additional encoders options, and take account of encoding instructions that specify variations of the BASIC-XER encodings in order to support specific styles of XML documents (see below). The extended XML Encoding Rules are not canonical, and there is no canonical form for these rules defined in this Recommendation | International Standard.

There are many aspects of an XML representation of data (such as the use of XML attributes instead of child elements, or the use of white-space delimited lists) whose use is a matter of style and XML designer choice. If a type defined in an ASN.1 specification is encoded by BASIC-XER or by CXER, then there is a single fixed style used for the XML representation, with no user control of stylistic features. This ITU-T Recommendation | International Standard specifies the syntax and semantics of XER encoding instructions which specify the stylistic features of the XML in an EXTENDED-XER encoding. XER encoding instructions can also be used to determine the possible insertion of XML processing instructions in an EXTENDED-XER encoding. XER encoding instructions are ignored by BASIC-XER and by CXER, but are used by EXTENDED-XER.

NOTE – "Stylistic features", such as use of attributes or white-space delimited lists, can also affect the size of an encoding and the ease with which it can be processed, so use of such features is not just a matter of style. Where such issues are important, EXTENDED-XER with encoding instructions may be preferred over BASIC-XER or CXER.

Clause 8 specifies the BASIC-XER encoding of ASN.1 types.

Clause 9 specifies the CXER encoding of ASN.1 types.

Clause 10 specifies the EXTENDED-XER encoding of ASN.1 types, referencing later clauses which define the XER encoding instructions.

Clauses 11 to 14 list and categorize the XER encoding instructions and specify the syntax for their assignment to an ASN.1 type or component using either an XER type prefix (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 30.3) or an XER encoding control section (see ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 50).

Clause 15 defines the order of precedence if XER encoding instructions are present in both an XER type prefix and in an XER encoding control section.

Clause 16 specifies the XER encoding instruction support for XML namespaces when using EXTENDED-XER.

Clause 17 specifies EXTENDED-XER encodings.

Clauses 18 to 39 specify:

a) the syntax of each XER encoding instruction used in a type prefix or an encoding control section;

b) restrictions on the XER encoding instructions that can be associated with a particular ASN.1 type (resulting from inheritance and multiple assignments);

c) modifications to the XER encoding rules that are required in an EXTENDED-XER encoding when an XER encoding instruction is applied.

Annex A is informative and contains examples of BASIC-XER and CXER encodings.

Annex B is informative and contains a description of the partial XML content that is produced when constructions such as sequence and sequence-of have their surrounding tags removed, together with restrictions on EXTENDED-XER specifications that enable easy determination of the ASN.1 component that an XML element is associated with.

Annex C is informative and contains examples of XER encoding instructions and of the corresponding EXTENDED-XER encodings.

Replace clause 1 with the following:

1 Scope

This Recommendation | International Standard specifies a set of basic XML Encoding Rules (BASIC-XER) that may be used to derive a transfer syntax for values of types defined in ITU-T Rec. X.680 | ISO/IEC 8824-1 and ITU-T Rec. X.681 | ISO/IEC 8824-2. This Recommendation | International Standard also specifies a set of Canonical XML Encoding Rules (CXER) which provide constraints on the basic XML Encoding Rules and produce a unique encoding for any given ASN.1 value. This Recommendation | International Standard further specifies a set of extended XML Encoding Rules (EXTENDED-XER) which adds further encoders options, and also allows the ASN.1 specifier to vary the encoding that would be produced by BASIC-XER. It is implicit in the specification of these encoding rules that they are also used for decoding.

The encoding rules specified in this Recommendation | International Standard:

– are used at the time of communication;

– are intended for use in circumstances where displaying of values and/or processing them using commonly available XML tools (such as browsers) is the major concern in the choice of encoding rules;

– allow the extension of an abstract syntax by addition of extra values for all forms of extensibility described in ITU-T Rec. X.680 | ISO/IEC 8824-1.

This Recommendation | International Standard also specifies the syntax and semantics of XER encoding instructions, and the rules for their assignment and combination. XER encoding instructions can be used to control the EXTENDED-XER encoding for specific ASN.1 types.

Replace subclause 2.1 with the following:

2.1 Identical Recommendations | International Standards

– 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.1 (2003) | ISO/IEC 8824-1:2002/Amd.1:2003, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation – Amendment 1: Support for EXTENDED-XER*.

– 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.681 (2002)/Amd.1 (2003) | ISO/IEC 8824-2:2002/Amd.1:2003, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification – Amendment 1: Support for EXTENDED-XER*.

– 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.1 (2003) | ISO/IEC 8825-1:2002/Amd.1:2003, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) – Amendment 1: Support for EXTENDED-XER*.

– 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.1 (2003) | ISO/IEC 8825-2:2002/Amd.1:2003, *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER) – Amendment 1: Support for EXTENDED-XER*.

– ITU-T Recommendation X.692 (2002) | ISO/IEC 8825-3:2002, *Information technology – ASN.1 encoding rules: Specification of Encoding Control Notation (ECN)*.

Replace subclause 2.2 with the following:

2.2 Additional references

– IETF RFC 2045 (1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies.*

– IETF RFC 2141 (1997), *URN Syntax.*

– IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax.*

– IETF RFC 3061 (2001), *A URN Namespace of Object Identifiers.*

– ISO/IEC 10646-1:2000, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane.*

– The Unicode Standard, Version 3.2.0, The Unicode Consortium. (Reading, MA, Addison-Wesley)

NOTE – The graphics characters (and their encodings) defined by the above reference are identical to those defined by ISO/IEC 10646-1, but the above reference is included because it also specifies the names of control characters.

– W3C XML 1.0:2000, *Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, Copyright © [6 October 2000] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/2000/REC-xml-20001006.*

– W3C XML Namespaces:1999, *Namespaces in XML, W3C Recommendation, Copyright © [14 January 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, Institut National de Recherche en Informatique et en Automatique, Keio University), http://www.w3.org/TR/1999/REC-xml-names-19990114.*

NOTE – The reference to a document within this Recommendation | International Standard does not give it, as a stand-alone document, the status of a Recommendation or International Standard.

Replace clause 3 with the following:

3 Definitions

For the purposes of this Recommendation | International Standard, the definitions of ITU-T Rec. X.680 | ISO/IEC 8824-1 and the following definitions apply.

3.1 ASN.1 Basic Encoding Rules (BER)

This Recommendation | International Standard makes use of the following terms defined in ITU-T Rec. X.690 | ISO/IEC 8825-1:

a) data value;

b) dynamic conformance;

c) encoding (of a data value);

d) receiver.

e) sender;

f) static conformance.

3.2 Additional definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

3.2.1 ASN.1 schema: The definition of the content and structure of data using an ASN.1 type definition.

NOTE – This enables encoding rules to produce binary encodings of the values of an ASN.1 type, or encodings using XML.

Insert four new subclauses 3.2.1 bis, 3.2.1 ter, 3.2.1 quat, and 3.2.1 quin as follows:

3.2.1 *bis* associated empty-element tag: The XML empty-element tag that can replace an associated preceding tag and an associated following tag, when present.

NOTE – Some encoding instructions remove the associated tags of an "XMLValue".

3.2.1 *ter* associated encoding instructions (for a type): A set of XER encoding instructions associated with a type.

3.2.1 *quat* associated following tag: The XML end-tag following the "XMLValue" of a type in the absence of encoding instructions that remove the associated tags.

3.2.1 *quin* associated preceding tag: The XML start-tag preceding the "XMLValue" of a type in the absence of encoding instructions that remove the associated tags.

Insert 16 new subclauses 3.2.2 bis through, 3.2.2 septdec as follows:

3.2.2 *bis* canonical valid XML document (for an ASN.1 schema): An XML document which is well-formed (see W3C XML 1.0) and whose content conforms to the CXER specification for the encoding of an ASN.1 type specified by an ASN.1 schema.

3.2.2 *ter* character-encodable type: An ASN.1 type to which an **ATTRIBUTE** encoding instruction can be applied (see 20.2.1).

3.2.2 *quat* control namespace: A namespace that is used to identify attributes that perform functions or carry values that control an EXTENDED-XER encoding.

NOTE 1 – An example would be a type identification attribute. The control namespace defaults to the ASN.1 namespace specified in 16.9, but can be changed by the **GLOBAL-DEFAULTS** encoding instruction.

NOTE 2 – The control namespace may also contain names for attributes that may be present, but which are ignored by EXTENDED-XER decoders (see 10.2.10). An example of such an attribute name could be **schemaLocation**.

3.2.2 *quin* enclosed (ASN.1) type: An ASN.1 type whose "XMLValue" in a BASIC-XER encoding is enclosed directly within the "XMLValue" of an ASN.1 type (the enclosing type).

NOTE – All types in a BASIC-XER or EXTENDED-XER encoding are enclosed types unless they are used as the root type (see 10.3.1 b) in an encoding.

3.2.2 *sex* enclosing element (of an ASN.1 type): An "ExtendedXMLTypedValue", "ExtendedXMLChoiceValue", "ExtendedXMLNamedValue" or "ExtendedXMLDelimitedItem" that has as its "ExtendedXMLValue" the "ExtendedXMLValue" encoding of the type (see 17.1, 17.5, 17.6 and 17.7).

3.2.2 *sept* enclosing type (of an ASN.1 type): An ASN.1 type whose "XMLValue" in a BASIC-XER encoding directly encloses the "XMLValue" of an ASN.1 type (an enclosed type).

NOTE – The enclosing type can be a sequence type, a set type, a choice type, a sequence-of type, a set-of type, an open type, or an octetstring or bitstring type (with a **CONTAINING** and without an **ENCODED BY**).

3.2.2 *oct* final encoding instructions (for a type): The set of XER encoding instructions associated with a type as a result of the complete ASN.1 specification, and which are applied in producing encodings of that type.

3.2.2 ***non*** **inherited encoding instructions**: XER encoding instructions that are associated with the type identified by a type reference.

3.2.2 ***dec*** **namespace-qualified name**: A name in an XML document that has an XML namespace prefix or is an XML element name in the scope of an XML default namespace declaration.

NOTE – XML default namespace declarations affect only XML element names, not the names of attributes. A namespace prefix can be applied to either.

3.2.2 ***unodec*** **nil identification attribute**: An XML attribute that can appear on any element to identify whether the content has a **nil** value (see clause 33).

3.2.2 ***duodec*** **partial XML element content**: XML child elements defined by an ASN.1 type which is **UNTAGGED**, and which provides part of the XML element content generated by the enclosing type.

NOTE – If the enclosing type is itself **UNTAGGED**, then that enclosing type may also be generating only partial XML element content.

3.2.2 ***tredec*** **prefixed encoding instructions**: XER encoding instructions that are assigned using a type prefix.

NOTE – Prefixed encoding instructions can delete, replace, or add to the associated encoding instructions of a type**.**

3.2.2 ***quatdec*** **qualifying information**: Information supplied as part of the specification of a target for the assignment of an encoding instruction that identifies specific values of the target type.

3.2.2 ***quindec*** **targeted encoding instructions**: XER encoding instructions that are assigned using a target list in an XER encoding control section.

NOTE – Targeted encoding instructions can delete, replace, or add to the associated encoding instructions of a type.

3.2.2 ***sexdec*** **type identification attribute**: An XML attribute that can appear on any element to identify the type of that element (see clause 37).

3.2.2 ***septdec*** **Uniform Resource Identifier (URI)**: A globally unambiguous identifier, assigned according to any one of a number of URI schemes, used to provide identification of namespaces in EXTENDED-XER encodings.

NOTE – The URI scheme used by default for ASN.1 enables an ASN.1 object identifier value to be used to identify namespaces (see 16.9 and 29.1.5).

Replace subclause 3.2.3 with the following:

3.2.3 valid XML document (for an ASN.1 schema): An XML document which is well-formed (see W3C XML 1.0) and whose content conforms to the BASIC-XER, CXER or EXTENDED-XER specification for the encoding of an ASN.1 type specified by an ASN.1 schema, possibly including XER encoding instructions.

Insert four new subclauses 3.2.3 bis through 3.2.3 quin as follows:

3.2.3 ***bis*** **XER encoding instructions**: Encoding instructions that are associated with an ASN.1 type (or with a component of an ASN.1 type) by assignment to that type (or component) in an XER type prefix (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 30.3) or an XER encoding control section (see ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 50).

3.2.3 ***ter*** **XML attribute**: Part of an EXTENDED-XER encoding consisting of an "XMLValue" enclosed in quotation marks or apostrophes, preceded by an (attribute) name and an equals sign.

3.2.3 ***quat*** **XML element**: Part of an XML document specified in W3C XML 1.0.

NOTE – An XML element is either an empty-element tag or starts with a start-tag and ends with an end-tag. Both the start-tag and the empty-element tag can contain attribute encodings.

3.2.3 ***quin*** **XML element name**: The lexical item following a "<" or "</" lexical item in the associated tags.

Replace subclause 3.2.4 with the following:

3.2.4 XML document: A sequence of characters which conforms to the W3C XML 1.0 definition of document.

Insert two new subclauses 3.2.5 and 3.2.6 as follows:

3.2.5 XML processing instruction: Part of an XML document which carries information concerning the processing of some or all of that document (see W3C XML 1.0).

NOTE – The processing instruction identifies the type of processing for which it is applicable, and is ignored in other processing. It could be used to identify a style-sheet that is to be applied if the document is presented for human viewing.

3.2.6 XML prolog: The initial part of an XML document (which does not carry information about the value of the ASN.1 type that has been encoded).

Replace clause 4 with the following:

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

ASN.1 Abstract Syntax Notation One
CXER Canonical XML Encoding Rules
PDU Protocol Data Unit
UCS Universal Multiple-Octet Coded Character Set
URI Uniform Resource Identifier
UTC Coordinated Universal Time
UTF-8 UCS Transformation Format, 8-bit form
XER XML Encoding Rules
XML Extensible Markup Language

Replace clause 5 with the following:

5 *This clause was removed by Amendment 1*

Replace subclauses 6.1, 6.2 and 6.3 with the following:

6.1 This Recommendation | International Standard specifies three sets of encoding rules:

– Basic XML Encoding Rules (BASIC-XER);
– Canonical XML Encoding Rules (CXER);
– Extended XML Encoding Rules (EXTENDED-XER).

6.2 The basic set of encoding rules specified in this Recommendation | International Standard is BASIC-XER, which does not in general produce a canonical encoding, and which does not provide any user control over the style of XML which is produced.

6.3 A second set of encoding rules specified in this Recommendation | International Standard is CXER, which produces encodings that are canonical. This is defined as a restriction of implementation-dependent choices in the BASIC-XER encoding.

NOTE 1 – Any implementation conforming to CXER for encoding is conformant to BASIC-XER for encoding. Any implementation conforming to BASIC-XER for decoding is conformant to CXER for decoding. Thus, encodings made according to CXER are encodings that are permitted by BASIC-XER.

NOTE 2 – CXER produces encodings that have applications when authenticators need to be applied to abstract values.

Insert a new subclause 6.3 bis as follows:

6.3 *bis* The third set of encoding rules specified in this Recommendation | International Standard is EXTENDED-XER. This is defined as variations of the BASIC-XER encodings specified by XER encoding instructions (see 6 *bis*) associated with an ASN.1 type. In the absence of XER encoding instructions, an EXTENDED-XER encoding differs from a BASIC-XER encoding only because it provides more encoders options.

Replace subclause 6.4 with the following:

6.4 If a type encoded with CXER contains **`EMBEDDED PDV`**, **`EXTERNAL`** or **`CHARACTER STRING`** types, then the outer encoding ceases to be canonical unless the encoding used for all the **`EMBEDDED PDV`**, **`EXTERNAL`** and **`CHARACTER STRING`** types is canonical.

Insert a new clause 6 bis as follows:

6 *bis* Encoding instructions specified by this Recommendation | International Standard

6 *bis*.1 This Recommendation | International Standard specifies the syntax and semantics of XER encoding instructions (see clauses 11 to 39). XER encoding instructions only affect EXTENDED-XER encodings.

6 *bis*.2 ASN.1 forms a basic XML schema notation. The ASN.1 schema is used to define the content and structure of data using ASN.1 and the BASIC-XER (and CXER) encoding rules. It can be used without XER encoding instructions.

6 *bis*.3 XER encoding instructions provide wider flexibility in the XML documents that can be specified.

6 *bis*.4 XER encoding instructions are assigned to ASN.1 type definitions or to type references using either or both of XER type prefixes (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 30.3) and an XER encoding control section (see ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 50). If encoding instructions are associated with a type definition, they are carried with the ASN.1 type (through its type reference) into other type definitions and other ASN.1 modules. When EXTENDED-XER encodes a type that has XER encoding instructions associated with some or all of its parts, those final encoding instructions are applied and modify the EXTENDED-XER encodings that are produced.

NOTE – The final encoding instructions are also used when performing validation and/or decoding of an EXTENDED-XER encoding.

Replace subclause 7.1 with the following:

7.1 Dynamic conformance for the basic XML Encoding Rules is specified by clause 8, and dynamic conformance for the Canonical XML Encoding Rules is specified by clause 9, and dynamic conformance for the extended XML Encoding Rules is specified by clause 10.

Replace subclause 7.3 with the following:

7.3 Alternative encodings are permitted by the basic XML Encoding Rules and by the extended XML Encoding Rules as an encoder's option. Decoders that claim conformance to BASIC-XER shall support all BASIC-XER encoding alternatives. Decoders that claim conformance to EXTENDED-XER shall support all EXTENDED-XER encoding alternatives.

NOTE – This clause applies whether or not there are any final encoding instructions.

Replace subclause 8.1 and its subclauses with the following:

8.1 Production of a complete BASIC-XER encoding

8.1.1 A conforming BASIC-XER encoding is a valid XML document which shall consist (in order) of:

a) an XML prolog (which may be empty) as specified in 8.2;

b) an XML document element which is the complete encoding of a value of a single ASN.1 type as specified in 8.3.

8.1.2 The specification in 8.2 to 8.6 completely defines the BASIC-XER encoding

NOTE – Other constructs of W3C XML 1.0, such as XML processing instructions, are not allowed by those subclauses, and are never produced by a conforming BASIC-XER encoder.

8.1.3 The XML document shall be encoded using UTF-8 to produce a string of octets which forms the encoding specified in this Recommendation | International Standard. The ASN.1 object identifier for these encoding rules is specified in clause 40.

8.1.4 Where this Recommendation | International Standard uses the term "white-space", this means one or more of the following characters of the Unicode Standard: HORIZONTAL TABULATION (9), LINE FEED (10), CARRIAGE RETURN (13), SPACE (32). The numbers in parentheses are the decimal value of the characters of the Unicode Standard. The number and choice of characters that constitutes "white-space" is an encoder's option.

Insert a new subclause 8.1.5 as follows:

8.1.5 Where this Recommendation | International Standard uses the term "white-space with escapes", this means one or more of the characters listed in 8.1.4, with an encoder's option to represent any of these characters with an escape sequence of the form "**&#*n*;**" or "**&#x*n*;**" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.8).

Replace subclause 8.3.1 with the following:

8.3.1 The XML document element shall be an "XMLTypedValue" as specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 15.2, with the changes and restrictions specified in the following subclauses of this clause 8.3.

Insert a new subclause 8.3.3 bis as follows:

8.3.1 *bis* All occurrences of "ExternalTypeReference" within the "XMLTypedValue" shall be replaced by the "typereference" in that "ExternalTypeReference".

Replace subclause 8.3.3 with the following:

8.3.3 Where ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.1.4, 11.11 and 11.13, permits the use of ASN.1 white-space between lexical items or in "xmlbstring" or in "xmlhstring", the characters used shall be restricted to the "white-space" specified in 8.1.4.

Insert a new subclause 8.3.3 bis as follows:

8.3.3 *bis* The "XMLBooleanValue" specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 17.3, shall only be "EmptyElementBoolean" and the "XMLSequenceOfValue" and "XMLSetOfValue" with a component that is a boolean type shall be "ValueList".

Replace subclause 8.3.4 with the following:

8.3.4 The "XMLIntegerValue" specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 18.9, shall only be "XMLSignedNumber".

Insert two new subclauses 8.3.4 bis and 8.3.4 ter as follows:

8.3.4 *bis* The "XMLEnumeratedValue" specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 19.8, shall only be "EmptyElementEnumerated" and the "XMLSequenceOfValue" and "XMLSetOfValue" with a component that is an enumerated type shall be "ValueList".

8.3.4 *ter* The "XMLSpecialRealValue" specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 20.6, shall only be "EmptyElementReal".

Insert a new subclause 8.5 as follows:

8.5 Encoding of the open type

Both alternatives of "XMLOpenTypeFieldVal" (see ITU-T Rec. X.681 | ISO/IEC 8824-2, 14.6) can be used.

NOTE – The use of the "xmlhstring" alternative of "XMLOpenTypeFieldVal" is not recommended in general, as there are no mechanisms to identify the encoding rules being used to produce the "xmlhstring" in an instance of an encoding. Cases where this alternative may be convenient are when the message being encoded in XER (e.g. for display purposes) is the result of a previous binary encoding and has not been completely decoded, or when there are bilateral agreements.

Insert a new subclause 8.6 as follows:

8.6 Decoding of types with extension markers

8.6.1 A BASIC-XER decoder shall accept as a valid XML document BASIC-XER encodings of types with extension markers in which unknown extensions are present.

8.6.2 Unknown extensions in a sequence or set type result in unexpected XML elements with names distinct from any of the names of the next expected XML element.

NOTE – There may be multiple names for a known following XML element when optionality is present, but the extension additions will always have names that differ from all of these.

8.6.3 Unknown extensions in a choice type result in a single unexpected XML element in place of an element corresponding to one of the known choices. It will always have a different XML element name from that of any XML element that encodes a known alternative of the choice type.

8.6.4 Unknown extensions in an enumerated type result in an XML element with an unexpected content, but with no unexpected XML elements.

8.6.5 Unknown extensions arising from relaxation of a subtype constraint result in an encoding that can be a valid encoding of any value of the unconstrained type. Such encodings can produce unexpected content, but no unexpected XML elements.

Replace subclause 9.1 with the following:

9.1 General rules for canonical XER

Replace subclause 9.1.2 with the following (retaining the NOTE):

9.1.2 All lexical items forming the "XMLTypedValue" shall have no "white-space" between them (see 8.3.3).

Replace subclause 9.3.1 with the following:

9.3.1 If the "XMLTypedValue" alternative of "XMLBitStringValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 21.9) can be used (as specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 21.10), then it shall be used. Otherwise, the "xmlbstring" alternative shall be used with all "white-space" removed (see 8.3.3).

Replace subclause 9.4 with the following:

9.4 Octetstring value

If the "XMLTypedValue" alternative of "XMLOctetStringValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 22.3) can be used (as specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 22.4), then it shall be used. Otherwise, the "xmlhstring" alternative shall be used with all "white-space" removed (see 8.3.3), and all letters in upper-case.

Replace subclause 9.6.1 with the following:

9.6.1 The set type shall have the elements in its "RootComponentTypeList" sorted into the canonical order specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 8.6, and additionally for the purposes of determining the order in which components are encoded when one or more component is a choice type with no ASN.1 tag, each such choice type is ordered as though it has a tag equal to that of the smallest tag in the "RootAlternativeTypeList" of that choice type or any such choice types nested within it.

Replace subclause 9.7.1 with the following:

9.7.1 The order of the elements of an "XMLSetOfValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 27.3) shall be determined by sorting the character strings which represent the CXER encoding for each element as specified in 9.7.2 and 9.7.3.

Insert a new subclause 9.12 as follows:

9.12 Open type value

The "xmlhstring" alternative of "XMLOpenTypeFieldVal" shall not be used (see 8.5).

Insert the following 30 new clauses 10 through 39 before the existing clause 10 (note that the existing clause 10 is renumbered as clause 40 by this amendment):

10 Extended XML encoding rules

10.1 General

10.1.1 The extended XML encoding rules (EXTENDED-XER) augment and modify BASIC-XER. They enable ASN.1 to define the form and content of a much wider range of XML documents.

10.1.2 EXTENDED-XER extends BASIC-XER in three main ways.

a) It provides additional encoder's options (for example, for the insertion of XML Processing Instructions or XML Comment, and for the use of identifiers for named bits in a bitstring value);

b) It specifies a set of encoding instructions that can be used to specify modification of the BASIC-XER encoding of an ASN.1 type, including an encoding instruction to use simple text rather than empty-element tags for boolean, integer (with named numbers), enumerated, special values of real, and bitstring (with named bits) types;

c) It requires decoders to ignore (in the absence of encoding instructions) attributes from the control namespace that are unknown (for example, a **schemaLocation** attribute), and some known attributes that other XML tools may insert which may have different values from those that a conforming encoder can insert (for example, use of a type identification attribute). (See 10.2.10.)

10.1.3 If an ASN.1 specification does not contain any XER encoding instructions, then every BASIC-XER encoding of any abstract value of an ASN.1 type is also an EXTENDED-XER encoding of the same abstract value of that type.

NOTE – The opposite is not true. Even in the absence of XER encoding instructions, there are EXTENDED-XER encodings that are not conforming to BASIC-XER encodings (see 10.1.2 a and 10.1.2 c).

10.1.4 All occurrences of ASN.1 "Type" notation have an associated set (possibly empty) of XER encoding instructions (the final associated encoding instructions). Encoding instructions are associated with a "Type" through:

a) (Inherited encoding instructions) the presence of associated encoding instructions on the "Type" used in the definition of a "typereference" used as a "Type"; and

b) (Targeted encoding instruction) assignment of one or more XER encoding instructions to an occurrence of "Type" using an XER encoding control section (see ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 50); and

NOTE – An ASN.1 module can contain only one XER encoding control section, and hence only one XER "EncodingInstructionAssignmentList" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 50.2)

c) (Prefixed encoding instructions) assignment of one or more XER encoding instructions to an occurrence of "Type" using XER type prefixes (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 30.3); and

d) (Import-list encoding instructions) assignment of one or more encoding instructions to all type references imported from an identified ASN.1 module.

10.1.5 The effect of assigning an XER encoding instruction is to add, delete, or replace associated encoding instructions (see clause 15 for the rules that apply for multiple assignments of XER encoding instructions).

10.1.6 The order (or manner) in which encoding instructions become part of (or are removed from) the set of associated encoding instructions is not significant in the application of the final encoding instructions.

10.1.7 The final encoding instructions affect the EXTENDED-XER encoding of types. They have no other impact, and in particular are not associated with any value reference defined using the type, nor do they affect value mappings, nor do they affect other encoding rules.

NOTE – There are, however, requirements on the unambiguity of names that are affected by the presence of a final **NAME**, **NAMESPACE**, or **UNTAGGED** encoding instruction. These requirements can be interpreted either as restrictions on the way in which types with such final encoding instructions can be used, or as restrictions on the use of these encoding instructions.

10.2 EXTENDED-XER conformance

10.2.1 If an ASN.1 specification assigns XER encoding instructions in accordance with clauses 11 to 17 such that an ASN.1 type or component has final encoding instructions that violate the restrictions specified in clauses 18 onwards, then that ASN.1 specification is not in conformity with this Recommendation | International Standard, even if (without the XER encoding instructions) it would conform to all the requirements of ITU-T Rec. X.680 | ISO/IEC 8824-1.

NOTE – It is only occasionally illegal to assign an encoding instruction to a "Type", as it can be negated (removed from the set of associated encoding instructions) by a further assignment. It is the final encoding instructions that normally determine conformity of the specification. In some (but not all) cases, a final encoding instruction that is not applicable to the type to which it is being applied is ignored. If the clauses specifying the syntax and application of encoding instructions identify circumstances where an encoding instruction is ignored in the application of the final encoding instructions, then clauses specifying encodings do not normally mention the possible presence of that final encoding instruction.

10.2.2 A conforming EXTENDED-XER encoding of an ASN.1 type with no final encoding instructions shall be the encoding produced by the basic XML encoding rules (BASIC-XER) specified in clause 8, with the additional encoder's options specified in 10.2.5 and 10.2.6.

NOTE – EXTENDED-XER decoders are required by 10.2.4 to accept and process W3C XML document type declarations, but these are not generated by conforming encoders, and do not form part of EXTENDED-XER encodings.

10.2.3 The EXTENDED-XER encoding of an ASN.1 type with final encoding instructions, or with components (at any depth, and after resolving all type references) that have associated encoding instructions, shall be the encoding specified in clause 17.

NOTE – The final encoding instructions are applied in an EXTENDED-XER encoding, and are also used by decoders and validators of EXTENDED-XER encodings.

10.2.4 EXTENDED-XER decoders (whether **MODIFIED-ENCODINGS** was used or not – see clause 26) shall process any document type declaration (see W3C XML 1.0, 2.8) that is present, in accordance with the requirements for non-validating XML processors (see W3C XML 1.0, 5.1). This processing shall be performed (conceptually) before applying all other decoding requirements in this Recommendation | International Standard. EXTENDED-XER encoders shall not include a document type declaration.

10.2.5 An EXTENDED-XER encoder can (as an encoder's option) insert XML Processing Instructions or XML Comment (in addition to any that might be required by clause 30) in the XML document element or XML prolog in any position permitted by W3C XML 1.0. The syntactic form and semantics of XML Processing Instructions is specified in W3C XML 1.0, 2.6. The syntactic form and semantics of XML Comment is specified in W3C XML 1.0, 2.5.

10.2.6 If there is no **GLOBAL-DEFAULTS** encoding instruction with a **MODIFIED-ENCODINGS** keyword (see clause 26) in the XER Encoding Control Section, then:

a) the "XMLIntegerValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 18.9) may be either "XMLSignedNumber" or "EmptyElementInteger", as an encoder's option; and

b) the "XMLBitStringValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 21.9) may be any of the alternatives of this production, as an encoder's option. If the "XMLIdentifierList" is used, it shall be the "EmptyElementList".

10.2.7 If there is a **GLOBAL-DEFAULTS** encoding instruction with a **MODIFIED-ENCODINGS** keyword (see clause 26) in the XER Encoding Control Section, then:

a) the "XMLBooleanValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 17.3) shall be "TextBoolean"; and

b) the "ExtendedXMLIntegerValue" (see 17.4) shall be the "ModifiedXMLIntegerValue" alternative, defined in 17.8; and

NOTE – This allows the use of a text value for "NamedNumber"s of an integer type, as an encoder's option, but also modifies the syntax for numeric encodings of an integer value.

c) the "ExtendedXMLEnumeratedValue" (see 34.3) shall not be "EmptyElementEnumerated"; and

NOTE – In the absence of a **GLOBAL-DEFAULTS** of **MODIFIED-ENCODINGS**, it cannot be "TextEnumerated" (see 8.3.4 *bis* and 34.3).

d) the "ExtendedXMLRealValue" (see 17.4) shall be the "ModifiedXMLRealValue" alternative, defined in 17.9; and

e) the "XMLSpecialRealValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 20.6) shall be the "TextReal" alternative; and

f) the alternative of "XMLIdentifierList" in the "XMLBitStringValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 21.9) shall be "TextList" (see 10.2.8 b); and

g) the "XMLSequenceOfValue" and "XMLSetOfValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 25.3 and 27.3) shall be "XMLDelimitedItem" for all component types, with Table 5 ignored (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 25.5); and

h) the "xmlhstring" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.13) shall not contain "white-space" (see 8.1.4); and

i) all occurrences of "white-space" that is either outside XML tags or inside the values of XML attributes can be "white-space with escapes" (see 8.1.5) as an encoder's option.

NOTE – There are some encoding instructions (such as **UNTAGGED**) that cannot be used unless there is a **GLOBAL-DEFAULTS** of **MODIFIED-ENCODINGS**.

10.2.8 If a **GLOBAL-DEFAULTS** encoding instruction with a **MODIFIED-ENCODINGS** keyword (see clause 26) is present in the XER Encoding Control Section, then an **EXTENDED-XER** encoder can (as an encoder's option):

a) use the "TextInteger" alternative of "ModifiedXMLIntegerValue" (see 17.8), provided there is a "NamedNumber" for the integer value in the type definition (see also 10.2.7 b);

NOTE – The use of this encoding with named values that have been added in a later version can make the abstract value represented unreadable by an implementation of an earlier version of the specification.

b) use "XMLIdentifierList" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 21.9) for an "XMLBitStringValue", provided the bitstring value to be encoded does not contain any "one" bits that are not named bits (see also 10.2.7 f);

NOTE – The use of this encoding with named bits that have been added in a later version can make the abstract value represented unreadable by an implementation of an earlier version of the specification.

10.2.9 Where encoder's options are permitted in an EXTENDED-XER encoding, conforming decoders and validators shall accept all options.

10.2.10 Conforming decoders and validators shall accept, but may ignore, the presence of an attribute from the control namespace in any XML element of an encoding unless its presence and use is as specified in clauses 37 and 38. Encoders shall not generate such attributes except as specified in clauses 37 and 38.

NOTE – Other XML tools may insert such attributes. In general, an EXTENDED-XER decoder cannot easily determine the permitted value and meaning of some attributes from the control namespace. Their presence and value may be of use to an application if (for example) unexpected XML child elements are present that are (as a decoder's option) passed to the application – rather than being ignored or producing a fatal decoding error.

10.2.11 An ASN.1 specification is illegal unless it is possible, for all abstract values, for a decoder to determine unambiguously (using only the name of the XML tag and the contents of any previous XML element) the ASN.1 component (or extension marker) that an XML element is associated with.

NOTE 1 – The association cannot depend on the content of the XML element, or on its attributes, or on any subsequent XML element.

NOTE 2 – This condition is always satisfied when there are no XER encoding instructions, but the inappropriate use of **UNTAGGED** to remove associated tags round (for example) repetitions (sequence-of or set-of) and alternatives (choice) and the inappropriate use of **NAME** can result in illegal specifications.

NOTE 3 – Subclause 10.2.11 is a necessary condition for valid encodings, but it is recognized that it is in general not possible for an ASN.1 tool (or for a human author) to check for legality based on this top-level statement alone. Annex B provides a model of the effect of the use of **UNTAGGED**, and rules that, if followed, can ensure legality of the specification as defined in 10.2.11.

10.2.12 If an ASN.1 specification contains "ObjectClassFieldType"s that are open types (see ITU-T Rec. X.681 | ISO/IEC 8824-2, 14.2), with table constraints or type constraints, such constraints shall all be ignored in applying 10.2.11.

10.3 Structure of an EXTENDED-XER encoding

10.3.1 A complete EXTENDED-XER encoding is a well-formed XML document consisting (in order) of:

a) an XML prolog (which may be empty as an encoder's option) as specified in 8.2; and

b) an XML document element which is the complete encoding of a value of a single ASN.1 type, called the root type, as specified in clause 17.

10.3.2 The "XMLValue" encodings used for BASIC-XER encodings are modified for EXTENDED-XER encodings by the final encoding instructions for the "Type"s that they encode, and the final encoding instructions for their components (to any depth), together with any **GLOBAL-DEFAULTS** encoding instructions.

NOTE – In an extreme case, the entire contents of the XML document element for a heavily nested ASN.1 structure can (through the use of the **UNTAGGED** encoding instruction) consist of nothing more than a linear sequence of XML elements, where only the root element has child elements. The use of **UNTAGGED** is restricted to ensure that all such resulting linear sequences of XML elements can be mapped without ambiguity to the components of an abstract value of the ASN.1 root type (see 10.2.11).

10.3.3 The XML document element in an EXTENDED-XER encoding consists of a single XML element that shall be an "ExtendedXMLTypedValue" for the type being encoded (the root type). It can include attributes in its start-tag or empty-element tag, and can have a content that includes both child elements (see W3C XML) and untagged text. The child elements may themselves have both attributes and a content that includes both child elements and untagged text.

10.3.4 The abstract values of the components of an enclosing type are encoded as "ExtendedXMLValue"s (see 17.4), possibly modified by encoding instructions applied to them or to their own components. These "ExtendedXMLValue"s:

a) can be preceded by an XML start-tag and followed by an XML end-tag (called the associated tags) to form an element within the "ExtendedXMLValue" of the enclosing type; or

b) can (by the use of an **UNTAGGED** encoding instruction on a type that is not character-encodable) form a partial XML element content for the "ExtendedXMLValue" of the enclosing type; or

NOTE – Annex B describes the result of applying **UNTAGGED** as the production of partial XML element content that can combine with other encodings to form the XML element content for some enclosing element whose type has not been **UNTAGGED**.

c) can (by the use of an **UNTAGGED** encoding instruction on a character-encodable type) form the complete "ExtendedXMLValue" of the component; or

d) can (by the use of an **ATTRIBUTE** encoding instruction on a character-encodable type) form the "CharacterEncodableValue" in the "QuotedValue" of an "Attribute" (see 20.3.3).

10.3.5 If an "ExtendedXMLValue" is empty, and its associated tags have not been removed by the use of an **UNTAGGED** encoding instruction, then the associated preceding and following tags can (as an encoder's option) be replaced with an XML empty-element tag (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 16.8). This is called the associated empty-element tag.

10.3.6 The transformation specified in 10.3.5 is performed conceptually after completion of the entire encoding process, and can be prevented by a **PI-OR-COMMENT** encoding instruction (see clause 30) producing one or more XML Processing Instruction or XML Comment elements between the start-tag and end-tag.

10.3.7 The associated preceding tag, the associated following tag, and the associated empty-element tag are jointly referred to as the associated tags. The XML element names in the associated tags are called the associated tag names, and are (in the absence of final **NAME** and **NAMESPACE** encoding instructions) identifiers, type reference names, or "xmlasn1typename"s (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.25).

11 Notation, character set and lexical items used in XER encoding instructions

11.1 The notation used in specifying the syntax of an "EncodingInstruction" in an XER type prefix (see clause 13), and in an "EncodingInstructionAssignmentList" in an XER encoding control section (see clause 14) is that defined by ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 5.

11.2 ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 10, applies to an XER "EncodingInstruction" and to an XER "EncodingInstructionAssignmentList".

NOTE – In particular, arbitrary ASN.1 white-space characters can appear between lexical items in both of these syntactic constructs unless the "&" notation is used (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 5.4).

11.3 The general rules specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.1, also apply to an XER "EncodingInstruction" and to an XER "EncodingInstructionAssignmentList".

NOTE – In particular, ASN.1 comment can be included wherever ASN.1 white-space is allowed, and requirements for the insertion of white-space or comment between lexical items that could otherwise be confused are those specified in ITU-T Rec. X.680 | ISO/IEC 8824-1.

11.4 The following lexical items are used in this Recommendation | International Standard:

comment	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.6)
cstring	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.14)
identifier	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.3)
modulereference	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.5)
number	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.8)
typereference	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.2)
"**{**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"**}**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"**.**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"**:**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"**,**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"**;**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"**@**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"*****"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"**::=**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.16)
"**<**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"**>**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.26)
"**</**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.21)
"**/>**"	(see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.22)

Additional lexical items ("modifiedXMLNumber" and "modifiedXMLRealNumber") are defined and used in 17.8.3 and in 17.9.

12 Keywords

12.1 The words specified in 12.3 and 12.4 below are used in either or both of XER "EncodingInstruction"s and XER "EncodingInstructionAssignmentList"s (in addition to some ASN.1 reserved words), and can appear in such syntactic constructs only with the meaning assigned to them in the following clauses of this Recommendation | International Standard, except as specified in 12.2.

12.2 Keywords are not reserved words, but if an ASN.1 "typereference" that is the same as a keyword listed in 12.3 is needed in an XER "EncodingInstruction" or an XER "EncodingInstructionAssignmentList", then the production "ModuleAndTypeReference" (see 14.2.2) shall be used.

12.3 The keywords are:

AFTER-TAG	**DEFAULT-FOR-EMPTY**	**REPLACE**
AFTER-VALUE	**ELEMENT**	**TEXT**
ANY-	**EMBED-VALUES**	**UNCAPITALIZED**
ATTRIBUTES	**GLOBAL-DEFAULTS**	**UNTAGGED**
ANY-ELEMENT	**IN**	**UPPERCASED**
AS	**LIST**	**USE-NIL**
ATTRIBUTE	**LOWERCASED**	**USE-NUMBER**
BASE64	**MODIFIED-ENCODINGS**	**USE-ORDER**
BEFORE-TAG	**NAME**	**USE-QNAME**
BEFORE-VALUE	**NAMESPACE**	**USE-TYPE**
CAPITALIZED	**NOT**	**USE-UNION**
COLLAPSE	**PI-OR-COMMENT**	**WHITESPACE**
CONTROL-NAMESPACE	**PREFIX**	
DECIMAL		

12.4 Additional keywords are used in the "BuiltInTypeName" production (see 14.2.3), but these are all ASN.1 reserved words (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.27) and can never be used in ASN.1 as a "typereference".

13 Assigning an XER encoding instruction to an ASN.1 type using a type prefix

13.1 Final encoding instructions for a type can:

a) require the use of alternatives of an "ExtendedXMLValue" that are not "XMLValue" alternatives for that type; or

NOTE – Alternatives of the "ExtendedXMLValue" production include both the (unchanged) "XMLValue" production alternatives used in BASIC-XER, and alternative productions selected by XER encoding instructions.

b) change the associated tag name, the "AttributeName", or the value of the type identification attribute for the encoding of that type; or

c) cause the "ExtendedXMLValue" of a component of an ASN.1 type to be inserted as the "CharacterEncodableValue" in the "QuotedValue" of an "Attribute" (see 20.3.3); or

d) specify the XML namespace name for type reference names and identifiers defined in an ASN.1 module and recommend a namespace prefix to be used with that namespace; or

e) specify when a namespace-qualified name (instead of an unqualified name) is to be used in an XML element or as the name of an XML attribute; or

f) specify the removal of the associated tags, generally resulting in either untagged text or in partial XML element content (which can be preceded or followed by other partial XML element content – see Annex B); or

g) specify the insertion of one or more XML Processing Instructions or XML Comments (see clause 30):

1) before the associated preceding tag or the associated empty-element tag; or

2) between the associated preceding tag and the "ExtendedXMLValue"; or

NOTE 1 – This prohibits the use of an associated empty-element tag.

3) between the "ExtendedXMLValue" and the associated following tag; or

NOTE 2 – This prohibits the use of an associated empty-element tag.

4) after the associated following tag.

NOTE 3 – All the above prohibit the use of **UNTAGGED** to remove the associated tags (see 30.2.2).

13.2 XER encoding instructions can be assigned to ASN.1 types using either the "EncodingInstruction" production in an XER type prefix or the "EncodingInstructionAssignmentList" production in an XER encoding control section. Assignment using a type prefix is specified in this clause. Assignment using an XER encoding control section is specified in clause 14.

NOTE – The effect of multiple assignments of encoding instructions of the same category is specified in clause 15.

13.3 The XER "EncodingInstruction" production is:

EncodingInstruction::=
PositiveInstruction
| NegatingInstruction

PositiveInstruction ::=
AnyAttributeInstruction
| AnyElementInstruction
| AttributeInstruction
| Base64Instruction
| DecimalInstruction
| DefaultForEmptyInstruction
| EmbedValuesInstruction
| GlobalDefaultsInstruction
| ListInstruction
| NameInstruction
| NamespaceInstruction
| PIOrCommentInstruction
| TextInstruction
| UntaggedInstruction
| UseNilInstruction
| UseNumberInstruction
| UseOrderInstruction
| UseQNameInstruction
| UseTypeInstruction
| UseUnionInstruction
| WhitespaceInstruction

NegatingInstruction ::=
NOT PositiveInstruction
| ElementInstruction

13.4 The "ElementInstruction" (see clause 24) is a strict synonym for **NOT UNTAGGED**, and is not discussed further in this clause.

NOTE 1 – The **ELEMENT** synonym is provided to avoid the double negative, and for human readability of specifications. It will normally be used (in opposition to the **ATTRIBUTE** encoding instruction) to identify the nature of top-level types in the ASN.1 module. Top-level types that have neither **ELEMENT** nor **ATTRIBUTE** final encoding instructions will be supporting types that do not directly correspond to XML attributes or elements, and will usually be **UNTAGGED**.

NOTE 2 – There is no negating encoding instruction for **ELEMENT**. An **ELEMENT** encoding instruction can be cancelled by a subsequent **UNTAGGED** encoding instruction, but such usage is not recommended.

13.5 Each use of a "PositiveInstruction" in an XER type prefix or in an encoding control section assigns that XER encoding instruction to the corresponding "Type".

13.6 If the "Type" in a "TypeAssignment" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 15.1) has final encoding instructions, all uses of the corresponding "typereference" (in the module containing the "TypeAssignment" or in some other module) inherit its final associated encoding instructions, except that any final **NAME** and **NAMESPACE** encoding instructions are not inherited.

NOTE – These two encoding instructions affect the XML name used in place of a type reference name. Where the type reference name is used to define the type in a type assignment or the type of a component, it is not appropriate to inherit such final encoding instructions from its definition.

13.7 An encoding instruction in a type prefix or in an encoding control section can be a positive instruction, used to add or to replace an encoding instruction (use of "PositiveInstruction"), or a negating instruction used to cancel (use of "NegatingInstruction") one or more associated encoding instructions.

13.8 XER encoding instructions consist of four parts (some of which may be empty):

a) **NOT,** indicating negation or removal of encoding instructions of a given category; and

NOTE 1 – This is present for negating instructions (except "ElementInstruction") and absent for positive instructions.

b) a keyword identifying the category of the encoding instruction; and

NOTE 2 – This is always present.

c) identification of a target list for the assignment of the encoding instruction (possibly with qualifying information restricting its application to a subset of the values of the type); and

NOTE 3 – When used in a type prefix, the target list is always the "empty" production, as the target for the assignment is always the type associated with the type prefix (see 13.12). The target list is also always "empty" for the **GLOBAL-DEFAULTS** encoding instruction.

d) syntax, specific to each encoding instruction category, providing details of the encoding instruction in that category.

NOTE 4 – When used in a negating instruction, this is always the "empty" production. It is also absent from some XER encoding instructions for which the keyword is a sufficient definition.

13.9 Some XER encoding instructions require the specification of the abstract value of a type. This specification uses the "Value" production (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 16.7). If a "valuereference" is used as "Value", then this "valuereference" shall be defined in (or imported into) the ASN.1 module containing the XER encoding instruction.

NOTE – This means that the value can be specified either directly using basic ASN.1 value notation, or by a value reference that was specified using either basic ASN.1 value notation or XML value notation.

13.10 Table 1 lists in column 1 the alternatives in the "PositiveInstruction" productions. Column 2 gives the clauses that specify the requirements for use of these encoding instructions and the modified encodings that they produce. Column 3 gives the category of the encoding instruction.

NOTE – These categories are introduced in order to provide clear statements on the result of multiple assignments of encoding instructions from the same category.

Table 1 – Encoding instructions and their defining clauses and categories

Encoding instruction	Defining clause	Category
AnyAttributesInstruction	Clause 18	Any-attributes instruction
AnyElementInstruction	Clause 19	Any-element instruction
AttributeInstruction	Clause 20	Attribute instruction
Base64Instruction	Clause 21	Base64 instruction
DecimalInstruction	Clause 22	Decimal instruction
DefaultForEmptyInstruction	Clause 23	Default-for-empty instruction
ElementInstruction	Clause 24	Element instruction
EmbedValuesInstruction	Clause 25	Embed-values instruction
GlobalDefaultsInstruction	Clause 26	Global-defaults instruction (but see 15.3)
ListInstruction	Clause 27	List instruction
NameInstruction	Clause 28	Name instruction (but see 15.3)
NamespaceInstruction	Clause 29	Namespace instruction
PiOrCommentInstruction	Clause 30	Pi-or-comment instruction (but see 15.3)
TextInstruction	Clause 31	Text instruction (but see 15.3)
UntaggedInstruction	Clause 32	Untagged instruction
UseNilInstruction	Clause 33	Use-nil instruction
UseNumberInstruction	Clause 34	Use-number instruction
UseOrderInstruction	Clause 35	Use-order instruction
UseQNameInstruction	Clause 36	Use-qname instruction
UseTypeInstruction	Clause 37	Use-type instruction
UseUnionInstruction	Clause 38	Use-union instruction
WhitespaceInstruction	Clause 39	Whitespace instruction

13.11 Each of the alternatives of the "PositiveInstruction" production is in a defined category of encoding instruction (or in some cases encompasses multiple categories). The category of each encoding instruction is specified in column 3 of Table 1 (but see also 15.3 for encoding instructions that encompass multiple categories).

NOTE – The categories of encoding instructions are used in 15.4 to determine the effect of multiple assignment of encoding instructions.

13.12 The "TargetList" in all "EncodingInstruction" constructions that appear in a type prefix shall be "empty" and the target shall be the "Type" associated with the type prefix.

13.13 A negating instruction is in the same category as the corresponding positive instruction.

13.14 An ASN.1 type can never have associated with it more than one XER encoding instruction of a given category (see 15.3 and 15.4), no matter how they are assigned. The result of multiple assignments of an XER encoding instruction of a given category is specified in clause 15.

14 Assigning an XER encoding instruction using an encoding control section

14.1 The encoding instruction assignment list

14.1.1 XER encoding instructions can be assigned to ASN.1 types using either the "EncodingInstruction" production in an XER type prefix or the "EncodingInstructionAssignmentList" production in an XER encoding control section. Assignment using a type prefix is specified in clause 13. Assignment using an XER encoding control section is specified in this clause.

14.1.2 The XER "EncodingInstructionAssignmentList" production is:

EncodingInstructionAssignmentList ::=
EncodingInstruction
EncodingInstructionAssignmentList ?

14.1.3 The "EncodingInstruction" production is defined in 13.3.

14.1.4 Each use of an "EncodingInstruction" in an encoding control section assigns that XER encoding instruction to the occurrences of "Type" that are identified in the "TargetList" of the encoding instruction, or to the type references in an imports list. The "TargetList" production and the targets it identifies are specified in 14.2.

14.1.5 Subclauses 13.4 to 13.14 also apply to encoding instructions in an encoding control section. The clauses defining the detailed syntax for each encoding instruction category are listed in Table 1. Categories of XER encoding instructions are also listed in Table 1.

14.2 Identification of the targets for an XER encoding instruction using a target list

14.2.1 General rules

14.2.1.1 The "EncodingInstruction" alternatives specify the XER encoding instruction that is being assigned, and the target(s) for that assignment within the ASN.1 module. All targets are an occurrence of the "Type" production within the ASN.1 module.

NOTE – Multiple targets, in the same or in different ASN.1 type assignments, can be specified. A target that is the entire module, or all occurrences within the module of a built-in type or constructor can also be specified. Thus (using an XER encoding control section) a single "EncodingInstruction" can be used to assign a particular XER encoding instruction to all the types in an ASN.1 module that require to have that encoding instruction assigned.

14.2.1.2 In identifying the target(s) for the assignment of an XER encoding instruction, the production "TargetList" is used. This is defined in the following subclauses.

NOTE 1 – The "TargetList" production is referenced in clauses 18 onwards.

NOTE 2 – The "TargetList" production has an "empty" alternative. This is the only permitted alternative if the "EncodingInstruction" is used in a type prefix (see 13.12). This subclause 14.2 considers only the use of the "TargetList" in an encoding control section.

14.2.1.3 The "TargetList" production is:

TargetList ::=
Targets "," +
| empty

Targets ::=
TypeIdentification
| BuiltInTypeIdentification
| IdentifiersInContext
| ImportedTypesIdentification

14.2.1.4 If the "TargetList" is a list of one or more "Targets" productions, then each of the "Targets" identifies one or more targets ("Type"s to which the encoding instruction is assigned), but can also provide qualifying information for the encoding instruction, restricting its application to encodings using a particular identifier in the target type definition, or to the use of the empty element tags for control characters specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.5.

NOTE – The qualifying information is only present if the target is a boolean, bitstring, enumerated, integer, or restricted character string type definition (see 14.2.2.9).

14.2.1.5 A "TargetList" of "empty" is permitted only in a type prefix (when it is the only permitted alternative) and in the **GLOBAL-DEFAULTS** encoding instruction. In a type prefix, it identifies the type associated with the prefix. In the **GLOBAL-DEFAULTS** encoding instruction, it identifies all "Type"s in the module.

14.2.1.6 The XER encoding instruction (possibly with associated qualifying information) is assigned to all the types identified by the "TargetList" as specified in 14.2.1.10 to 14.2.1.16.

NOTE – It would be unusual, but not illegal, for a given "Type" to be identified more than once in the target list. In such cases, clause 15 applies.

14.2.1.7 (Tutorial) Identification of the target(s) (and possible qualifying information) by the "Targets" production uses one of five basic forms:

a) use of a "typereference" (see 14.2.2), possibly followed by a dot-separated list of identifiers, identifying either:
   1) the "Type" in a type assignment (no identifiers present); or
   2) the "Type" in a component of a type definition (which can include top-level components introduced by the **COMPONENTS OF** construct – see 14.2.1.12); or
   3) one of 1) or 2), plus a final identifier (preceded by a colon, not a dot) for an identifier used in the target type definition, providing the qualifying information;

b) use of **ALL** as the last identifier in the a) form, identifying all of the "Type"s textually present in the type definition (that is identified by the preceding type reference and dot-separated list of identifiers), or qualifying information (preceded by a colon, not a dot) identifying all of the identifiers used for values of a boolean, bitstring, enumerated, or integer type definition (that is identified by the preceding type reference and dot-separated list of identifiers) or identifying all uses of the XML empty-element tags used to represent some control characters (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.5);

c) use of a "BuiltInTypeName" (see 14.2.3), identifying all "Type"s in the module that are defined by use of the corresponding built-in type name or constructor, possibly (in the case of **BOOLEAN**, **BIT STRING**, **ENUMERATED, INTEGER** and restricted character string types only) followed by qualifying information;

d) use of a list of "identifier"s followed by **IN** (or **ALL** followed by **IN,** or **COMPONENTS** followed by **IN)** and the a) form above (see also 14.2.4), identifying:
   1) the "Type" of the identified components of the a) form; or
   2) (use of **ALL**) all "Type"s that textually occur within the "Type" identified by the a) form; or
   3) (use of **COMPONENTS**) all "Type"s that are the top-level components of the "Type" identified by the a) form;

e) use of "ImportedTypesIdentification" (see 14.2.5) identifies all the "typereference"s in the **IMPORTS** list that are imported from a specified module.

NOTE 1 – The term "type definition" used in a) and b) above emphasizes that only textually present identifiers can be used. Identifiers cannot be used if the "Type" is a type reference.

NOTE 2 – In general a component can be referenced by use of a) or d) above. If more than one component of a type is to be referenced, then d) would be preferred as it is less verbose, otherwise a) would be preferred. This is a matter of style.

14.2.1.8 A bitstring or octetstring type with a contents constraint that contains a type shall be treated as a type with a single component, using "*****" as the component identifier, for the purpose of assigning a targeted instruction to the "Type" in the contents constraint.

14.2.1.9 A type definition that is a sequence-of or a set-of shall be treated as a type with a single component, using "*****" as the component identifier, for the purpose of assigning a targeted instruction to the "Type" that is the component of the sequence-of or set-of.

NOTE – It is also possible to identify this single component using the component identifier (if present).

14.2.1.10 If a target is the use of a dummy parameter of a parameterized type, the target inherits the final encoding instructions of the actual parameter before encoding instructions targeting the dummy parameter are assigned. The specification is legal only if the resulting final encoding instructions for all instantiations of the parameterized type are legal.

NOTE 1 – If the parameterized type is exported, the final encoding instructions for its dummy parameters are carried with it.

NOTE 2 – There are no mechanisms provided to assign encoding instructions directly to the "Type" of an actual parameter in an instantiation of a parameterized type.

14.2.1.11 If the target is a "SelectionType", the target inherits the final encoding instructions of the selected alternative of the choice type referenced by the selection type, after which encoding instructions assigned to the "SelectionType" are assigned.

14.2.1.12 If the target is a component produced as a result of the **COMPONENTS OF** transformation, the target inherits the final encoding instructions of the component of the type referenced by the **COMPONENTS OF,** after which encoding instructions assigned to the components produced by the **COMPONENTS OF** are assigned. Any encoding instructions for the "Type" from which the components are extracted are ignored.

14.2.1.13 If the "Targets" production is "TypeIdentification", then the targets it identifies are specified in 14.2.2.

14.2.1.14 If the "Targets" production is "BuiltInTypeIdentification", then the targets it identifies are specified in 14.2.3.

14.2.1.15 If the "Targets" production is "IdentifiersInContext", then the targets it identifies are specified in 14.2.4.

14.2.1.16 If the "Targets" production is "ImportedTypesIdentification", then the targets it identifies are specified in 14.2.5.

14.2.1.17 EXAMPLE: The example below shows an ASN.1 type definition followed by two different ways of assigning XER encoding instructions in an encoding control section, and finally, the same ASN.1 type definition with the XER encoding instructions assigned using type prefixes. All three approaches result in the same EXTENDED-XER encoding.

The type definition is:

```
My-Type ::= SEQUENCE {
      field1  INTEGER,
      field2  CHOICE {
                 first   SEQUENCE OF INTEGER,
                 second  SEQUENCE OF OBJECT IDENTIFIER } }
```

The XER encoding instructions in the encoding control section could be:

```
      ATTRIBUTE   field1 IN My-Type
      LIST        first IN My-Type.field2
```

Alternatively, they could be:

```
      ATTRIBUTE   My-Type.field1
      LIST        My-Type.field2.first
```

The type definition with type prefixes is:

```
My-Type ::= SEQUENCE {
      field1  [ATTRIBUTE] INTEGER,
      field2  CHOICE {
                 first   [LIST] SEQUENCE OF INTEGER,
                 second  SEQUENCE OF OBJECT IDENTIFIER } }
```

14.2.2 Target identification using an ASN.1 type reference and identifiers

14.2.2.1 The "TypeIdentification" production is:

```
TypeIdentification ::=
    ALL
    | ModuleAndTypeReference
```

```
        ComponentReference ?
        QualifyingInformationPart ?

    ModuleAndTypeReference ::=
        typereference
        | modulereference "." typereference

    ComponentReference ::=
            "."
            ComponentIdList

    ComponentIdList ::=
            ComponentId "." +

    ComponentId ::=
            identifier
            | "*"
            | ALL

    QualifyingInformationPart ::=
            ":"
            QualifyingInformation

    QualifyingInformation
            identifier
            | ALL
```

14.2.2.2 A "TypeIdentification" of **ALL** identifies all "Type"s in "TypeAssignment"s in the module.

14.2.2.3 The "ModuleAndTypeReference" production identifies the "Type" that is assigned to the "typereference". The "modulereference" in "ModuleAndTypeReference" shall be the module reference for the module containing the "EncodingInstructionAssignmentList", and the "typereference" shall be a type reference that is defined in the module. It shall be used if and only if the "typereference" consists of the same characters as one of the keywords specified in 12.3, otherwise the "typereference" alone shall be used.

14.2.2.4 A symbol"*****" identifies the "Type" of the (sole) component of a sequence-of or set-of type, or the type in a contents constraint that contains a "Type".

NOTE – This form can be used even if the sequence-of or set-of component has an identifier, but the use of the identifier should be preferred.

14.2.2.5 If **ALL** is used as a "ComponentId", it shall be the last "ComponentId" in the "ComponentIdList" and shall not be followed by "QualifyingInformation".

14.2.2.6 If the first "ComponentId" in the "ComponentIdList" (if present) is an identifier that is textually present (or results from use of **COMPONENTS OF**) as a component identifier in the "Type" identified by the "ModuleAndTypeReference", then it identifies the "Type" of that component. If it is not an identifier that is textually present (or results from use of **COMPONENTS OF**) as a component identifier in the "Type" identified by the "ModuleAndTypeReference", then this occurrence of "TypeIdentification" is not illegal, but does not identify any target.

NOTE – This requires that the type referenced by the "ModuleAndTypeReference" be a sequence, set, choice, sequence-of or set-of type definition, or a bitstring or an octetstring type definition with a contents constraint that contains a "Type".

14.2.2.7 If a subsequent "ComponentId" (except the last) in the "ComponentIdList" (if present) is an identifier that is textually present as a component identifier in the "Type" identified by the previous "ComponentId", then it identifies the "Type" of that component. If it is not a component identifier that is textually present in the "Type" identified by the previous "ComponentId", then this occurrence of "TypeIdentification" is not illegal, but does not identify any target.

NOTE – The first use of "ComponentId" can refer to components introduced by a **COMPONENTS OF**. Components of those components cannot be identified by subsequent "ComponentId"s.

14.2.2.8 If the last "ComponentId" in the "ComponentIdList" (if present) is:

a) an identifier that is textually present as a component identifier in the "Type" identified by the previous "ComponentId", then it identifies the "Type" of that component; the encoding instruction shall be assigned to that "Type"; or

b) the keyword **ALL**; the encoding instruction shall be assigned to all "Type"s that are textually present in the type definition identified by the previous "ComponentId", which shall be a type with one or more components.

14.2.2.9 The "QualifyingInformationPart" shall not be present unless the "ModuleAndTypeReference" with the "ComponentReference" (if present) identifies target(s) that are:

a) boolean types; or

b) bitstring types with named bits; or

c) enumerated types; or

d) integer types with named numbers; or

e) restricted character string types.

14.2.2.10 The "identifier" alternative of "QualifyingInformation" shall not be used unless the "ModuleAndTypeReference" with the "ComponentReference" (if present) identifies a single target that is not a restricted character string type, or identifies a target list all of whose types are the boolean type. The "identifier" shall be an identifier in the target type definition if the target is not a boolean type, or shall be **`true`** or **`false`**. The "identifier" is qualifying information that identifies that the encoding instruction applies only to encodings using that identifier.

14.2.2.11 The **`true`** and **`false`** alternatives of "QualifyingInformation" for a boolean type specify qualifying information that identifies that the encoding instruction applies only to the encoding of the **`TRUE`** or the **`FALSE`** abstract values, respectively.

14.2.2.12 The **`ALL`** alternative of "QualifyingInformation" shall not be used unless the target identifies (only) one or more type definitions for the types listed in 14.2.2.9. It shall not be used if the target identifies one or more restricted character string target(s), unless the encoding instruction being applied is **`NAMESPACE`**. It specifies qualifying information that identifies that the encoding instruction applies to all the identifiers in the type definitions, or in the case of a restricted character string type, to all uses of the XML empty-element tags used to represent the control characters listed in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.5.

NOTE – It is not possible to use qualifying information with an "identifier" to selectively affect the representation of control characters. Only **`ALL`** is possible in this case.

14.2.3 Target identification using a built-in type name

14.2.3.1 The "BuiltInTypeIdentification" production is:

```
BuiltInTypeIdentification ::=
    BuiltInTypeName
    BuiltInTypeQualifyingInformationPart ?

BuiltInTypeName::=
    BIT STRING
    | BOOLEAN
    | CHARACTER STRING
    | CHOICE
    | EMBEDDED PDV
    | ENUMERATED
    | EXTERNAL
    | GeneralizedTime
    | INSTANCE OF
    | INTEGER
    | NULL
    | ObjectDescriptor
    | OBJECT IDENTIFIER
    | OCTET STRING
    | REAL
    | RELATIVE-OID
    | SEQUENCE
    | SEQUENCE OF
    | SET
    | SET OF
    | UTCTime
    | RestrictedCharacterStringType

BuiltInTypeQualifyingInformationPart ::=
    ":"
    BuiltInTypeQualifyingInformation
```

BuiltInTypeQualifyingInformation
identifier
| ALL

14.2.3.2 The "BuiltInTypeIdentification" production specifies that the encoding instruction is to be applied to all textual occurrences within the module of the corresponding built-in type or of a type defined using the corresponding constructor.

14.2.3.3 The "RestrictedCharacterStringType" is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 37.

14.2.3.4 The "BuiltInTypeQualifyingInformationPart" shall not be present unless the "BuiltInTypeName" is **BOOLEAN**, **BIT STRING, ENUMERATED**, **INTEGER**, or a restricted character string type.

NOTE – Only the **ALL** form of "BuiltInTypeQualifyingInformation" is permitted for a restricted character string type (see 14.2.2.10 and the next subclause).

14.2.3.5 The "identifier" alternative of "BuiltInTypeQualifyingInformation" shall not be used unless the "BuiltInTypeName" is **BOOLEAN**, and shall then be either **true** or **false**. It specifies qualifying information that identifies that the encoding instruction applies only to the encoding of the **TRUE** or the **FALSE** abstract values, respectively.

14.2.3.6 The **ALL** alternative of "BuiltInQualifyingInformation" specifies qualifying information that identifies that the encoding instruction applies to all identifiers used in any instance of use of the "BuiltInTypeName" within the module (or to all values of the **BOOLEAN** type definition, or to all the empty-element tags used in values of the specified restricted character string type – see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.5).

14.2.4 Use of identifiers in context

14.2.4.1 The "IdentifiersInContext" production is:

IdentifiersInContext ::=
IdentifierList
IN
TypeIdentification

IdentifierList ::=
identifier "," +
| ALL
| COMPONENTS

14.2.4.2 "TypeIdentification" is defined in 14.2.2, and identifies a type defined in a type assignment statement in the module, or a component or sub-component of a type defined in the module. The "QualifyingInformationPart" shall be absent.

14.2.4.3 The "Type" identified by the "TypeIdentification" shall be a sequence, set or choice type, and is called for the purposes of this clause the identified "Type".

NOTE – The "TypeIdentification" in "IdentifiersInContext" cannot be used for a sequence-of or set-of type. Such use is prohibited for clarity, as it would be no less verbose than direct use of "TypeIdentification" in "Targets".

14.2.4.4 Each "identifier" in "IdentifierList" shall be the "identifier" of a component of the identified "Type". The XER encoding instruction is assigned to the "Type" of all the components of the identified "Type" that have a component "identifier" in the "IdentifierList".

14.2.4.5 The use of **ALL** for "IdentifierList" specifies that all textually present components (and all textually present components of those components, to any depth) in the identified "Type" are targets to which the XER encoding instruction is being assigned.

14.2.4.6 The use of **COMPONENTS** for "IdentifierList" specifies that all components (at the first level) of the identified "Type" are targets to which the XER encoding instruction is being assigned.

14.2.5 Use of imported types identification

14.2.5.1 The "ImportedTypesIdentification" production is:

ImportedTypesIdentification ::=
ALL IMPORTS FROM modulereference

14.2.5.2 The "modulereference" shall be one of the "modulereference"s used in one of the "GlobalModuleReferences" of the imports clause of the module.

14.2.5.3 The XER encoding instruction is assigned to each of the "typereference"s in the corresponding "SymbolList", after the final encoding instructions produced by assignment in the exporting module have been assigned.

14.2.5.4 If an imported "typereference" is exported from this module, the final encoding instructions inherited by that "typereference" in a module that imports it are those inherited in this importing module, and are not affected by assignment of encoding instructions using an "ImportedTypesIdentification". This assignment affects only the use of the type reference within this module.

15 Multiple assignment of XER encoding instructions

15.1 Order in which multiple assignments are considered

15.1.1 A "Type" which is not a "typereference" has initially an empty set of associated encoding instructions.

15.1.2 A "Type" which is a "typereference" (which may be imported) has initially the set of final encoding instructions of the "Type" which was assigned to it when it was defined (possibly modified by encoding instructions assigned to it in the imports list of an importing module – see 14.2.5).

15.1.3 Targeted encoding instructions for a "Type" (using an encoding control section) are assigned next, in the order in which the targeted encoding instructions appear in the encoding control section. If the "Type" is identified by more than one element of a "TargetList" (see 14.2), then that shall be treated as multiple assignments of the same encoding instruction to that "Type", in the order in which the elements occur in the "TargetList".

NOTE – The effect of 15.1.2 and 15.1.3 means that targeted assignment to a "Type" in a "TypeAssignment" is always over-ridden by a targeted assignment to a "Type" defined using the corresponding "typereference", no matter which targeted assignment appears first in the encoding control section. However, if a targeted assignment is made to all the components of a type, and also to an individual component of that type, the effect will depend on the order of the encoding instructions in the encoding control section.

15.1.4 Prefixed encoding instructions (using a type prefix) assigned to a type are considered next, with the rightmost (the innermost) prefixed encoding instruction considered first, and the leftmost (the outermost) prefixed encoding instruction considered last.

15.1.5 As specified in 14.2.1.10, encoding instructions are assigned to a dummy parameter only after the final encoding instructions for the actual parameter have been determined.

15.1.6 As specified in 14.2.1.11 and 14.2.1.12, a "SelectionType" and the components produced by a **COMPONENTS OF** transformation inherit first the final encoding instructions of the original type, and then have encoding instructions targeted at them applied.

15.1.7 Each assignment of an encoding instruction produces a new set of associated encoding instructions, as specified in 15.2 to 15.4.

15.2 Effect of assigning a negating encoding instruction

15.2.1 All assignments of a negating encoding instruction result in the removal (from the set of associated encoding instructions) of any encoding instruction of the same category. If there are no associated encoding instructions of a different category, the set becomes empty.

15.2.2 The **NOT GLOBAL-DEFAULTS** encoding instruction shall never be assigned.

15.2.3 For those encoding instructions with multiple categories (see 15.3), a negating encoding instruction removes all the encoding instructions in any of those categories.

NOTE – A negating encoding instruction never becomes part of the set of associated encoding instructions.

15.3 Multiple assignment of encoding instructions with multiple categories

15.3.1 The **NAME** and **TEXT** encoding instructions (see clauses 28 and 31) can be assigned to a type to either:

a) change the associated tag name (no "QualifyingInformation" present); or

NOTE – This applies only to the **NAME** encoding instruction.

b) change the "ExtendedXMLValue" encoding by providing a new name for a specified "identifier" present in the type definition ("QualifyingInformation" present that is not **ALL**); or

c) change the "ExtendedXMLValue" encoding by providing a modification to be applied to all "identifier"s present in the type definition ("QualifyingInformation" present that is **ALL**, with a target that is not a restricted character string type).

15.3.2 In case 15.3.1 b), the encoding instruction for a specified "identifier" is treated as a different category from an encoding instruction for any other "identifier", and from an encoding instruction for 15.3.1 a).

15.3.3 In case 15.3.1 c), the encoding instruction is expanded into a set of encoding instructions of type 15.3.1 b), with one encoding instruction for each "identifier" present in the type definition.

15.3.4 The **PI-OR-COMMENT** encoding instruction (see clause 30) has four categories, corresponding to the four alternatives for "Position".

15.3.5 Subject to 15.3.3 to 15.3.4, subclause 15.4 specifies the rules for multiple assignment of XER encoding instructions.

15.3.6 Each of the alternatives of the **GLOBAL-DEFAULTS** encoding instruction is a separate category, but each category of this encoding instruction shall be assigned at most once.

15.4 Multiple assignment of XER encoding instructions of the same category

NOTE – Multiple assignment of XER encoding instructions of the same category is expected to be rare, except where an XML encoding instruction is assigned globally, and an overriding (possibly negating) encoding instruction is assigned to specific types or components. This subclause specifies the rules if multiple assignment of XER encoding instructions in the same category occurs. This clause is also referenced by 15.3.5 for the treatment of multiple assignments of **NAME**, **PI-OR-COMMENT**, and **TEXT** encoding instructions.

15.4.1 Assignments of positive encoding instructions result in the addition (to the set of associated encoding instructions) of that XER encoding instruction if there are no other associated encoding instructions of the same category.

15.4.2 Assignment of an **ELEMENT** encoding instruction is always equivalent to assignment of a **NOT UNTAGGED** encoding instruction.

15.4.3 If there is an encoding instruction of the same category in the set of associated encoding instructions, then that encoding instruction is removed from the set, and the assigned XER encoding instruction is added.

NOTE – If encoding instructions are being assigned globally in an encoding control section, with the intention of overriding them in specific cases, then the overriding has to be done using either a type prefix or a later encoding instruction in the encoding control section, not an earlier one.

15.4.4 If a type that appears in a "ContentsConstraint" or in a "TypeConstraint" is to be encoded by EXTENDED-XER, then the final encoding instructions (as determined by the above rules) are used in determining the encoding of that type. If a type appears in any other ASN.1 constraint, then all associated encoding instructions are discarded.

15.5 Permitted combinations of final encoding instructions

15.5.1 Table 2 specifies the permitted combinations of final encoding instructions for a "Type" when a **GLOBAL-DEFAULTS** of **MODIFIED-ENCODINGS** has been used. Column 1 lists all encoding instructions. Column 2 lists all the encoding instructions that can be used in combination with the column 1 encoding instruction as final encoding instructions, but in many cases restrictions apply that are listed in the applicable clauses.

NOTE – **GLOBAL-DEFAULTS** is not listed in the table, as this is not assigned to a type.

Table 2 – Permitted combinations of final encoding instructions with MODIFIED-ENCODINGS

Encoding instruction	Permitted other encoding instructions
ANY-ATTRIBUTES *(see clause 18)*	**ELEMENT, NAME, NAMESPACE**
ANY-ELEMENT *(see clause 19)*	**ELEMENT, NAME, NAMESPACE**
ATTRIBUTE *(see clause 20)*	**BASE64, DECIMAL, ELEMENT, LIST, NAME, NAMESPACE, TEXT, USE-NUMBER, USE-QNAME, USE-UNION, WHITESPACE**
BASE64 *(see clause 21)*	**ATTRIBUTE, DEFAULT-FOR-EMPTY, ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT, UNTAGGED**
DECIMAL *(see clause 22)*	**ATTRIBUTE, DEFAULT-FOR-EMPTY, ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT, UNTAGGED**
DEFAULT-FOR-EMPTY *(see clause 23)*	**BASE64, DECIMAL, ELEMENT, EMBED-VALUES, LIST, NAME, NAMESPACE, PI-OR-COMMENT, TEXT, USE-NIL, USE-NUMBER, USE-ORDER, USE-QNAME, USE-UNION, WHITESPACE**
ELEMENT *(see clause 24)*	*Equivalent to* **NOT UNTAGGED**
EMBED-VALUES *(see clause 25)*	**DEFAULT-FOR-EMPTY, ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT, USE-NIL, USE-ORDER**
LIST *(see clause 27)*	**ATTRIBUTE, DEFAULT-FOR-EMPTY, ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT, UNTAGGED**
NAME *(see clause 28)*	*No restrictions*
NAMESPACE *(see clause 29)*	*No restrictions*
PI-OR-COMMENT *(see clause 30)*	**BASE64, DECIMAL, DEFAULT-FOR-EMPTY, ELEMENT, EMBED-VALUES, LIST, NAME, NAMESPACE, TEXT, USE-NIL, USE-NUMBER, USE-ORDER, USE-QNAME, USE-TYPE, USE-UNION, WHITESPACE**
TEXT *(see clause 31)*	**ATTRIBUTE, DEFAULT-FOR-EMPTY, ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT, UNTAGGED**
UNTAGGED *(see clause 32)*	**BASE64, DECIMAL, LIST, NAME, NAMESPACE, TEXT, USE-NUMBER, USE-QNAME, USE-UNION, WHITESPACE**
USE-NIL *(see clause 33)*	**DEFAULT-FOR-EMPTY, ELEMENT, EMBED-VALUES, NAME, NAMESPACE, PI-OR-COMMENT, USE-ORDER**
USE-NUMBER *(see clause 34)*	**ATTRIBUTE, DEFAULT-FOR-EMPTY, ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT, UNTAGGED**
USE-ORDER *(see clause 35)*	**DEFAULT-FOR-EMPTY, ELEMENT, EMBED-VALUES, NAME, NAMESPACE, PI-OR-COMMENT, USE-NIL**.
USE-QNAME *(see clause 36)*	**ATTRIBUTE, DEFAULT-FOR-EMPTY, ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT, UNTAGGED**
USE-TYPE (*see clause 37)*	**ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT**
USE-UNION *(see clause 38*)	**ATTRIBUTE, DEFAULT-FOR-EMPTY, ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT, UNTAGGED**
WHITESPACE *(see clause 39)*	**ATTRIBUTE, DEFAULT-FOR-EMPTY, ELEMENT, NAME, NAMESPACE, PI-OR-COMMENT, UNTAGGED**

15.5.2 Table 3 specifies the permitted combinations of final encoding instructions when a **GLOBAL-DEFAULTS** of **MODIFIED-ENCODINGS** has not been used. Column 1 lists all encoding instructions that are permitted as final encoding instructions if a **GLOBAL-DEFAULTS** of **MODIFIED-ENCODINGS** has not been used. Column 2 either says "*Not permitted*" or lists all the encoding instructions that can be used in combination with the column 1 encoding instruction as final encoding instructions, but in many cases restrictions apply that are listed in the applicable clauses. "*Not permitted*" means that that encoding instruction cannot be used as a final encoding instruction if a **GLOBAL-DEFAULTS** of **MODIFIED-ENCODINGS** has not been used.

NOTE – **GLOBAL-DEFAULTS** is not listed in the table, as this is not assigned to a "Type".

Table 3 – Permitted combinations of final encoding instructions with no MODIFIED-ENCODINGS

Encoding instruction	Permitted other encoding instructions
ANY-ATTRIBUTES	*Not permitted*
ANY-ELEMENT	*Not permitted*
ATTRIBUTE	**BASE64, LIST, NAME, TEXT, USE-NUMBER, WHITESPACE**
BASE64	**ATTRIBUTE, NAME, PI-OR-COMMENT**
DECIMAL	*Not permitted*
DEFAULT-FOR-EMPTY	*Not permitted*
ELEMENT	*Not permitted*
EMBED-VALUES	*Not permitted*
LIST	**ATTRIBUTE, NAME, PI-OR-COMMENT**
NAME	**ATTRIBUTE, BASE64, LIST, PI-OR-COMMENT, TEXT, USE-NUMBER, WHITESPACE**
NAMESPACE	*Not permitted*
PI-OR-COMMENT	**BASE64, LIST, NAME, TEXT, USE-NUMBER, WHITESPACE**
TEXT	**ATTRIBUTE, NAME, PI-OR-COMMENT**
UNTAGGED	*Not permitted*
USE-NIL	*Not permitted*
USE-NUMBER	**ATTRIBUTE, NAME, PI-OR-COMMENT**
USE-ORDER	*Not permitted*
USE-QNAME	*Not permitted*
USE-TYPE	*Not permitted*
USE-UNION	*Not permitted*
WHITESPACE	**ATTRIBUTE, NAME, PI-OR-COMMENT**

16 XER encoding instruction support for XML namespaces and qualified names

16.1 W3C XML Namespaces defines concepts and rules governing necessary qualifiers and mechanisms to ensure that an XML element name or attribute name can be correctly identified with a corresponding specification of the associated semantics.

16.2 W3C XML Namespaces defines an XML namespace as a collection of unambiguous names, identified by a URI, which are used in XML documents as element types and attribute names. The URI that identifies a namespace is called the namespace name. In this Recommendation | International Standard, namespaces are also used to qualify the values of a type that has a final encoding instruction of **USE-QNAME** (see clause 36) and that represents an XML QName (see W3C XML Schema, Part 2, 3.2.18).

16.3 Type reference names and identifiers can (but need not) be assigned a namespace.

NOTE – This Recommendation | International Standard uses a namespace name that is, by default, a form of URI based on ASN.1 object identifiers (see clause 29). All other forms of URI can be used to assign a namespace name to the names in an ASN.1 module.

16.4 Whether or not a type is part of an XML namespace (and if so its namespace name) is determined by the presence (or absence) of a final **NAMESPACE** encoding instruction.

NOTE – A **NAMESPACE** encoding instruction can only be present if a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction is also present in the encoding control section (see 29.2.1).

16.5 A namespace is identified by the "NameSpaceSpecification" production that provides the Uniform Resource Identifier for the namespace, and optionally a recommended namespace prefix. The "NameSpaceSpecification" is specified in clause 29.

16.6 Names of XML elements and attributes in an EXTENDED-XER encoding are generated from several sources. The subclauses of 16.8 list the sources of XML element and attribute names, identify what namespace they are part of, and specify whether they are to be namespace-qualified names or not.

16.7 An XML element name, an XML attribute name, or a value of a type identification attribute may (but need not) have a final **NAMESPACE** encoding instruction on the "Type" that generates the name. If it does, then the name shall be a namespace-qualified name in the encoding. (The namespace-qualification in an encoding can be done either explicitly using a defined XML namespace prefix, or indirectly by establishing a default XML namespace for a scope that includes the use of the name or of the value.) If there is no **NAMESPACE** encoding instruction on a "Type" that generates a name, then the name is not a namespace-qualified name. Names that are not namespace-qualified names are called unqualified names, and shall not occur in the scope of an established default XML namespace.

NOTE – BASIC-XER does not support XML namespaces, and namespace-qualified names never occur in BASIC-XER encodings.

16.8 In the following subclauses, the term "ASN.1 namespace" refers to the namespace whose name and recommended prefix are specified in 16.9. The term "assigned namespace" refers to the namespace assigned by the **NAMESPACE** encoding instruction to a type. If generated names are not from the ASN.1 namespace, and there is no such assignment of a namespace name, then the XML element names, XML attribute names, and values of type identification attributes are unqualified names.

16.8.1 In all the subclauses of this 16.8, the element names and attribute names in the XML tags (whether XML empty-element tags or start tags) are namespace-qualified names in an encoding if and only if the generating "Type" has a final **NAMESPACE** encoding instruction.

16.8.2 Element names in XML empty-element tags used for control characters (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.5) have no namespace unless one is assigned by the application of a **NAMESPACE** encoding instruction to the restricted character string type with qualifying information of **ALL**.

16.8.3 Element names in XML empty-element tags used for values of the integer, enumerated, bitstring types, and special values of real types (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 18.9, 19.8, 20.6 and 21.9) will always be unqualified names (see 16.7) in an encoding of these types.

16.9 The namespace of the type identification attribute (see clause 37) and of the nil identification attribute (see clause 33) is the control namespace, which is, by default, the ASN.1 namespace, unless a different control namespace is specified by a **GLOBAL-DEFAULTS** encoding instruction (see clause 26). The ASN.1 namespace has a name of **"urn:oid:2.1.5.2.0.1"** (see 40.3), and a recommended namespace prefix of **"asn1"**. (See also 26.3.2.)

16.10 For an octetstring type with a contents constraint that specifies an EXTENDED-XER encoding, any abstract value of the octetstring type shall be a complete EXTENDED-XER encoding of a value of an ASN.1 type (see ITU-T Rec. X.682 | ISO/IEC 8824-3, 11.5 and 11.6), and shall contain all necessary namespace declarations for all prefixed and unprefixed qualified names present in the octetstring abstract value.

NOTE – Such an octetstring type is encoded as an "xmlhstring" or "Base64OctetStringValue". Any namespace declarations present in the XML document that contains the "xmlhstring" or "Base64OctetStringValue" do not include in their scope the names present in the octet string.

16.11 When an open type is encoded as an "xmlhstring" or "Base64XMLOpenTypeFieldVal", and the encoding rules used for the contained type are EXTENDED-XER, the "xmlhstring" or the "Base64XMLOpenTypeFieldVal" shall be the hexadecimal or base64 representation (respectively) of an octet string that is a complete EXTENDED-XER encoding of a value of the contained type, and shall contain all necessary namespace declarations for all prefixed and unprefixed qualified names present in it.

NOTE – Any namespace declarations present in the XML document that contains the "xmlhstring" or the "Base64XMLOpenTypeFieldVal" do not include in their scope the names present in the octet string.

17 Specification of EXTENDED-XER encodings

The specification of EXTENDED-XER encodings uses the productions specified in the following subclauses. These productions allow all of the syntax of the corresponding productions used by BASIC-XER (of the same name but with "Extended" removed), but provide additional syntax that is allowed in EXTENDED-XER encodings. The use of this additional syntax is determined by the application of XER encoding instructions, and is specified in clauses 18 to 39.

NOTE – The alternative productions available are frequently restricted by the use or non-use of a **GLOBAL-DEFAULTS** encoding instruction with the **MODIFIED-ENCODINGS** keyword (see 10.2.7 and 10.2.8). In particular, the use of empty-element or text encodings for some built-in types is controlled by this.

17.1 The XML document element

17.1.1 The XML document element shall be an "ExtendedXMLTypedValue".

17.1.2 "ExtendedXMLTypedValue" is:

ExtendedXMLTypedValue ::=
"<" & TypeNameOrModifiedTypeName AttributeList ">"
ExtendedXMLValue
"</" & TypeNameOrModifiedTypeName ">"
| "<" & TypeNameOrModifiedTypeName "/>"

NOTE – The differences from the "XMLTypedValue" production are the inclusion of a possibly empty "AttributeList", and the use of an "ExtendedXMLValue" instead of an "XMLValue" for the contents of the XML element.

17.1.3 "TypeNameOrModifiedTypeName" is defined in 17.2.

17.1.4 "AttributeList" is defined in 17.3.

17.1.5 "ExtendedXMLValue" is defined in 17.4, and shall be the "ExtendedXMLValue" of the type identified by the "TypeNameOrModifiedTypeName".

17.1.6 The second alternative of "XMLTypedValue" (use of an XML empty-element tag) can be used only if an instance of the "ExtendedXMLValue" production is empty.

NOTE – If the "ExtendedXMLValue" production was an "xmlcstring" containing only "white-space", this would not be empty, and the second alternative could not be used.

17.2 The "TypeNameOrModifiedTypeName" production

17.2.1 "TypeNameOrModifiedTypeName" is:

TypeNameOrModifiedTypeName ::=
NonParameterizedTypeName
| QualifiedOrUnqualifiedName

17.2.2 "NonParameterizedTypeName" is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, 13.2, and is used (as specified in that subclause and in ITU-T Rec. X.680 | ISO/IEC 8824-1, 13.4 to 13.7) as an XML element name that identifies an ASN.1 type.

17.2.3 "QualifiedOrUnqualifiedName" is specified in 29.3.2. The "QualifiedOrUnqualifiedName" alternative shall be used if and only if there is a final **NAME** or **NAMESPACE** encoding instruction applied to the type (see clause 28), otherwise the "NonParameterizedTypeName" shall be used.

17.3 The "AttributeList" production

17.3.1 The "AttributeList" is:

AttributeList ::=
Attribute AttributeList
| empty

17.3.2 The "Attribute" is defined in 20.3.3.

17.3.3 The "AttributeList" shall be empty unless the application of final encoding instructions requires its use (see clauses 20, 33, and 37).

17.3.4 The "Attribute"s in the "AttributeList" shall be preceded by "white-space" (see 8.1.4).

17.4 The "ExtendedXMLValue" production

17.4.1 "ExtendedXMLValue" is:

ExtendedXMLValue ::=
ExtendedXMLBuiltinValue
| ExtendedXMLObjectClassFieldValue
| empty

```
ExtendedXMLBuiltinValue ::=
    XMLBitStringValue
    | XMLBooleanValue
    | ExtendedXMLCharacterStringValue
    | ExtendedXMLChoiceValue
    | XMLEmbeddedPDVValue
    | ExtendedXMLEnumeratedValue
    | XMLExternalValue
    | XMLInstanceOfValue
    | ExtendedXMLIntegerValue
    | XMLNullValue
    | XMLObjectIdentifierValue
    | ExtendedXMLOctetStringValue
    | ExtendedXMLRealValue
    | XMLRelativeOIDValue
    | ExtendedXMLSequenceValue
    | ExtendedXMLSequenceOfValue
    | ExtendedXMLSetValue
    | ExtendedXMLSetOfValue
    | ExtendedXMLPrefixedValue

ExtendedXMLCharacterStringValue ::=
    ExtendedXMLRestrictedCharacterStringValue
    | XMLUnrestrictedCharacterStringValue

ExtendedXMLRestrictedCharacterStringValue ::=
    XMLRestrictedCharacterStringValue
    | Base64XMLRestrictedCharacterStringValue

ExtendedXMLObjectClassFieldValue ::=
    ExtendedXMLOpenTypeFieldVal
    | XMLFixedTypeFieldVal

ExtendedXMLOpenTypeFieldVal ::=
    ExtendedXMLTypedValue
    | Base64XMLOpenTypeFieldVal
    | xmlhstring

ExtendedXMLOctetStringValue ::=
    ExtendedXMLTypedValue
    | Base64XMLOctetStringValue
    | xmlhstring

ExtendedXMLRealValue ::=
    XMLRealValue
    | ModifiedXMLRealValue

ExtendedXMLIntegerValue ::=
    XMLIntegerValue
    | ModifiedXMLIntegerValue

ExtendedXMLPrefixedValue ::=
    ExtendedXMLValue
```

17.4.2 The alternatives of "ExtendedXMLBuiltinValue" whose production names do not start with "Extended", and their use to encode abstract values, are fully specified in ITU-T Rec. X.680 | ISO/IEC 8824-1 (see 16.10 and 16.2 of that Recommendation | International Standard) and (for "XMLFixedTypeFieldVal" and the third alternative of "ExtendedXMLOpenTypeFieldVal") in ITU-T Rec. X.681 | ISO/IEC 8824-2, 14.6.

17.4.3 The "Base64XMLRestrictedCharacterStringValue" is defined in 21.3.5 and shall only be used as specified in that subclause.

17.4.4 The "ExtendedXMLChoiceValue" is defined in 17.5 and shall only be used as specified in that subclause.

17.4.5 The "ExtendedXMLEnumeratedValue" is defined in 34.3 and shall only be used as specified in that subclause.

17.4.6 The "ExtendedXMLSequenceValue" and "ExtendedXMLSetValue" are defined in 17.6 and shall only be used as specified in that subclause.

17.4.7 The "ExtendedXMLSequenceOfValue" and "ExtendedXMLSetOfValue" are defined in 17.7 and shall only be used as specified in that subclause.

17.4.8 The "Base64XMLOctetStringValue" and "Base64XMLOpenTypeFieldVal" are defined in 21.3.2 and 21.3.4 and shall only be used as specified in those subclauses.

17.4.9 The "ModifiedXMLIntegerValue" is defined in 17.8 and shall only be used as specified in that subclause.

17.4.10 The "ModifiedXMLRealValue" is defined in 17.9 and shall only be used as specified in that subclause.

17.4.11 The "empty" alternative of "ExtendedXMLValue" shall only be used as specified in clause 23.

NOTE – The other alternatives of "ExtendedXMLValue" can also produce an "empty" lexical item. This subclause does not affect the use of such occurrences.

17.5 The "ExtendedXMLChoiceValue" production

17.5.1 The "ExtendedXMLChoiceValue" is:

```
ExtendedXMLChoiceValue ::=
    "<" &  TagName AttributeList ">"
    ExtendedXMLValue
    "</" & TagName ">"
    | ExtendedXMLValue

TagName ::=
    IdentifierOrModifiedIdentifier

IdentifierOrModifiedIdentifier ::=
    identifier
    | QualifiedOrUnqualifiedName
```

17.5.2 The "QualifiedOrUnqualifiedName" is defined in 29.3.2. The "QualifiedOrUnqualifiedName" shall be used if and only if there is a final **NAME** encoding instruction (see clause 28), or a final **NAMESPACE** encoding instruction applied to the type (see clause 29), otherwise the "identifier" shall be used.

NOTE – If "identifier" is used, then the encoding cannot include an XML default namespace declaration with a scope that includes the use of that "identifier" (see 16.7).

17.5.3 The "AttributeList" and its use is defined in 17.3 and the clauses it references.

17.5.4 The "ExtendedXMLValue" in both alternatives of the "ExtendedXMLChoiceValue" shall be the "ExtendedXMLValue" of the selected alternative of the choice type.

17.5.5 The second alternative of "ExtendedXMLChoiceValue" shall be used if either:

a) the selected alternative of the choice type has an **UNTAGGED** final encoding instruction (see clause 32); or

b) the choice type has a **USE-TYPE** or **USE-UNION** final encoding instruction (see clauses 37 and 38).

NOTE – This means that the presence of these final encoding instructions results in the omission of XML tags as a choice determinant, and choice determination has to occur by other means (see clauses 37, 38 and Annex B).

17.6 The "ExtendedXMLSequenceValue" and "ExtendedXMLSetValue" productions

17.6.1 The "ExtendedXMLSequenceValue" and "ExtendedXMLSetValue" are:

```
ExtendedXMLSequenceValue ::=
    ExtendedXMLComponentValueList
    | empty
```

ExtendedXMLSetValue ::=
ExtendedXMLComponentValueList
| empty

ExtendedXMLComponentValueList ::=
ExtendedXMLNamedValue
| ExtendedXMLComponentValueList ExtendedXMLNamedValue

ExtendedXMLNamedValue ::=
"<" & TagName AttributeList ">"
ExtendedXMLValue
"</" & TagName ">"
| ExtendedXMLValue

17.6.2 The "empty" alternatives of "ExtendedXMLSequenceValue" and "ExtendedXMLSetValue" shall only be used if no component of the sequence or set type (to any depth), after resolution of all type references and after application of all final encoding instructions, produces an "ExtendedXMLNamedValue".

NOTE – This includes (but is not limited to) the cases in which all components are marked **DEFAULT** or **OPTIONAL** and all values are omitted; have a final **UNTAGGED** encoding instruction and their values have an empty encoding; have a final **ATTRIBUTE** encoding instruction. It also includes combinations of the above, and the case in which the type notation is **SEQUENCE {}** or **SET {}**.

17.6.3 The "TagName" is defined in 17.5.1. The "QualifiedOrUnqualifiedName" in the "IdentifierOrModifiedIdentifier" form of "TagName" shall be used if and only if there is a final **NAME** or **NAMESPACE** encoding instruction applied to the type (see clause 29), otherwise the "identifier" shall be used.

17.6.4 The "AttributeList" and its use is defined in 17.3 and the clauses it references.

17.6.5 The "ExtendedXMLValue" in both alternatives of the "ExtendedXMLNamedValue" shall be the "ExtendedXMLValue" of the component of the sequence or set type.

17.6.6 The second alternative of "ExtendedXMLSequenceValue" and "ExtendedXMLSetValue" shall be used if and only if the alternative has an **UNTAGGED** final encoding instruction (see clause 32).

17.7 The "ExtendedXMLSequenceOfValue" and "ExtendedXMLSetOfValue" productions

17.7.1 The "ExtendedXMLSequenceOfValue" and "ExtendedXMLSetOfValue" are:

ExtendedXMLSequenceOfValue ::=
ExtendedXMLValueList
| ExtendedXMLDelimitedItemList
| empty
| ExtendedXMLListValue

ExtendedXMLSetOfValue ::=
ExtendedXMLValueList
| ExtendedXMLDelimitedItemList
| empty
| ExtendedXMLListValue

ExtendedXMLValueList ::=
ExtendedXMLValueOrEmpty
| ExtendedXMLValueOrEmpty ExtendedXMLValueList

ExtendedXMLValueOrEmpty ::=
ExtendedXMLValue
| "<" & TypeNameOrModifiedTypeName "/>"

ExtendedXMLDelimitedItemList ::=
ExtendedXMLDelimitedItem
| ExtendedXMLDelimitedItem ExtendedXMLDelimitedItemList

ExtendedXMLDelimitedItem ::=
"<" & TypeNameOrModifiedTypeName AttributeList ">"
ExtendedXMLValue
"</" & TypeNameOrModifiedTypeName ">"
| "<" & IdentifierOrModifiedIdentifier AttributeList ">"
ExtendedXMLValue
"</" & IdentifierOrModifiedIdentifier ">"
| ExtendedXMLValue

17.7.2 The use of the alternatives of "ExtendedXMLSequenceOfValue", "ExtendedXMLSetOfValue" and of "ExtendedXMLValueList" shall be in accordance with the use of the corresponding alternatives of "XMLSequenceOfValue", "XMLSetOfValue" and of "XMLValueList" (respectively) as specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, clauses 25 and 27, except that if a **GLOBAL-DEFAULTS** encoding instruction with a **MODIFIED-ENCODINGS** keyword is present, "ExtendedXMLValueList" shall never be used (see also 10.2.7 g).

17.7.3 The "ExtendedXMLListValue" is defined in 27.3.2. These alternatives of "ExtendedXMLSequenceOfValue" and "ExtendedXMLSetOfValue" shall be used only if there is a final **LIST** encoding instruction (see clause 27) on the sequence-of or set-of type.

17.7.4 The first alternative of the "ExtendedXMLDelimitedItem" shall be used if and only if the sequence-of or set-of type does not contain an "identifier" and the component does not have a final **UNTAGGED** encoding instruction. The following subclauses apply.

17.7.4.1 If the component of the sequence-of or set-of type is a "typereference" or an "ExternalTypeReference" (possibly with one or more "TypePrefix"s), then the "TypeNameOrModifiedTypeName" shall be the "typereference" or the "typereference" in the "ExternalTypeReference", respectively, possibly modified in accordance with any final **NAME** and **NAMESPACE** encoding instructions applied to the component (see clause 28).

17.7.4.2 If the component of the sequence-of or set-of type (after ignoring any occurrences of "TypePrefix") is not a "typereference" or an "ExternalTypeReference", then the "TypeNameOrModifiedTypeName" shall be the "xmlasn1typename" specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, Table 4, corresponding to the built-in type of the component, possibly modified in accordance with any final **NAMESPACE** encoding instruction applied to the component (see clause 29).

17.7.5 The second alternative of the "ExtendedXMLDelimitedItem" shall be used if and only if the sequence-of or set-of type contains an "identifier" and the component does not have a final **UNTAGGED** encoding instruction. The "IdentifierOrModifiedIdentifier" shall be that "identifier", possibly modified in accordance with any final **NAME** and **NAMESPACE** encoding instructions applied to the component (see clauses 28 and 29).

17.7.6 The third alternative of "ExtendedXMLDelimitedItem" shall be used if and only if the component of the sequence-of or set-of type has a final **UNTAGGED** encoding instruction (see clause 32).

17.7.7 The "ExtendedXMLValue" in all the alternatives of the "ExtendedXMLDelimitedItem" shall be the "ExtendedXMLValue" of the repeated component of the sequence-of or set-of type.

17.7.8 The "TypeNameOrModifiedTypeName" in the "ExtendedXMLValueOrEmpty" shall be the "xmlasn1typename" specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, Table 4, corresponding to the built-in type of the component, possibly modified in accordance with any final **NAMESPACE** encoding instruction applied to the component (see clause 29).

17.8 The "ModifiedXMLIntegerValue" production

17.8.1 The "ModifiedXMLIntegerValue" is:

ModifiedXMLIntegerValue ::=
ModifiedXMLSignedNumber
| TextInteger

ModifiedXMLSignedNumber ::=
modifiedXMLNumber
| "-" & modifiedXMLNumber
| "+" & modifiedXMLNumber

17.8.2 This alternative of "ExtendedXMLIntegerValue" (see 17.4) shall only be used if a **GLOBAL-DEFAULTS** encoding instruction with a **MODIFIED-ENCODINGS** keyword is assigned.

17.8.3 The "modifiedXMLNumber" lexical item shall consist of one or more digits.

NOTE 1 – The "modifiedXMLnumber" lexical item is mapped to an integer value by interpreting it as decimal notation.

NOTE 2 – This lexical item differs from "number" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.8) lexical item, only because it permits any number of leading "0" digits.

17.8.4 Any positive integer value can be encoded using either the first or the third alternative of "ModifiedXMLSignedNumber", as an encoder's option. A negative integer value shall be encoded using the second alternative. The integer value zero can be encoded using any of the three alternatives, as an encoder's option.

17.8.5 "TextInteger" is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, 18.9, and provides an alternative encoding (as an encoder's option) for integer values that have a "NamedNumber" definition.

17.9 The "ModifiedXMLRealValue" production

17.9.1 The "ModifiedXMLRealValue" is:

ModifiedXMLRealValue ::=
ModifiedXMLNumericRealValue
| XMLSpecialRealValue
| XMLDecimalMinusZeroRealValue

ModifiedXMLNumericRealValue ::=
modifiedXMLRealNumber
| "-" & modifiedXMLRealNumber
| "+" & modifiedXMLRealNumber

17.9.2 This alternative of "ExtendedXMLRealValue" (see 17.4) shall only be used if a **GLOBAL-DEFAULTS** encoding instruction with a **MODIFIED-ENCODINGS** keyword is assigned.

17.9.3 The "modifiedXMLRealNumber" lexical item shall consist of an integer part that is a series of one or more digits, and optionally a decimal point (.). The decimal point can optionally be followed by a fractional part that is one or more digits. The integer part, decimal point or fractional part (whichever is last present) can optionally be followed by an e or E and an optionally-signed exponent which is one or more digits.

NOTE – This lexical item differs from the "realnumber" (see ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.9) lexical item only because it permits any number of leading zeros in the exponent.

17.9.4 Any positive real value and the real value plus zero can be encoded using either the first or the third alternative of "ModifiedXMLNumericRealValue", as an encoder's option. Any negative real value shall be encoded using the second alternative of "ModifiedXMLNumericRealValue". The real value minus zero shall be encoded using the second alternative.

17.9.5 The "XMLDecimalMinusZeroRealValue" is defined in 22.3.2 and shall only be used as specified in that subclause.

NOTE – The **DECIMAL** encoding instruction defined in 22.3.2 provides this production as an alternative representation for the positive zero abstract real value, but requires that the minus zero abstract value be excluded from the type to which it is applied.

18 The ANY-ATTRIBUTES encoding instruction

18.1 General

18.1.1 The "AnyAttributesInstruction" is:

AnyAttributesInstruction ::=
ANY-ATTRIBUTES
TargetList
NamespaceRestriction ?

NamespaceRestriction ::=
FROM URIList
| EXCEPT URIList

URIList ::=
QuotedURIorAbsent
| URIList QuotedURIorAbsent

QuotedURIorAbsent ::=
QuotedURI
| ABSENT

18.1.2 The "TargetList" production is defined in 14.2.

18.1.3 The "QuotedURI" is defined in 29.1.1.

18.1.4 This encoding instruction is assigned to an ASN.1 type that is a sequence-of or set-of type with a **UTF8String** component whose value provides zero, one or more attribute names and values (one in each **UTF8String**), each of which is subject to any "NamespaceRestriction" that is present (see 18.2).

NOTE – Although sequence-of may be used for the specification of the attributes, this use of sequence-of does not imply that order is semantically significant, and the encoding/decoding process may result in a different order of the components of the sequence-of.

18.1.5 The content of each **UTF8String** is encoded as an "Attribute" of the enclosing XML element. The name of the sequence-of or set-of component is ignored.

18.1.6 The **FROM** and **EXCEPT** clauses (if present) identify lists of namespace names, or the special keyword **ABSENT**.

18.1.7 **FROM** restricts attribute names to be namespace-qualified names from one of the specified namespaces. If **ABSENT** is present in the "URIList", unqualified names can also be used.

18.1.8 **EXCEPT** allows namespace-qualified names from any namespaces except those listed. It also allows unqualified names unless **ABSENT** is present in the "URIList".

18.2 Restrictions

18.2.1 An ASN.1 type shall not have this final encoding instruction unless it is a set-of or sequence-of type with a component that is a **UTF8String** type.

18.2.2 A type with this final encoding instruction shall only be used as a component of an enclosing sequence or set type, and the component shall not be marked **OPTIONAL** or **DEFAULT**. There shall only be one such component in the enclosing type.

18.2.3 A sequence-of or set-of type with this final encoding instruction is required to have a constraint applied to it that imposes the format and content specified in 18.2.6 to 18.2.11 on each occurrence of the **UTF8String**, by reference to this clause 18 or otherwise.

NOTE – It is recommended that the constraint on the sequence-of or set-of type be expressed as:

```
(CONSTRAINED BY
  {/* Each UTF8String shall conform to the "AnyAttributeFormat" specified in
      ITU-T Rec. X.693 |  ISO/IEC 8825-4, clause 18. */})
```

18.2.4 There shall be no final **UNTAGGED** encoding instruction on either the type that has this final encoding instruction or on the enclosing type.

18.2.5 Each "URIList" shall contain at most one occurrence of **ABSENT** and shall not contain two identical "QuotedURI"s.

18.2.6 The format of each **UTF8String** shall conform to the production "AnyAttributeFormat":

AnyAttributeFormat ::=
URI ?
NCName & "=" & xmlcstring

18.2.7 See 29.1.4 for the definition of the "URI" production, and 29.1.7 for the definition of the "NCName" production. The "xmlcstring" lexical item is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.

18.2.8 If there is a "NamespaceRestriction" of **FROM**, then the "URI" in "AnyAttributeFormat" shall be the "URI" in a "QuotedURI" in the "URIList", and may be absent only if the keyword **ABSENT** occurs in the "URIList".

18.2.9 If there is a "NamespaceRestriction" of **EXCEPT**, then the "URI" in "AnyAttributeFormat" shall not be the "URI" in a "QuotedURI" in the "URIList", and shall not be absent if the keyword **ABSENT** occurs in the "URIList".

18.2.10 The "xmlcstring" shall be a syntactically correct XML attribute value (defined in W3C XML, clause 3) preceded and followed by either a single APOSTROPHE (39) character or by a single QUOTATION MARK (34) character.

18.2.11 Application of this encoding instruction and the **ATTRIBUTE** encoding instruction to different components of the enclosing type shall not violate 20.3.11.

18.2.12 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULT MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

18.2.13 A type with this final encoding instruction shall not also have any of the final encoding instructions **LIST**, **PI-OR-COMMENT** or **UNTAGGED**.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ELEMENT**, **ATTRIBUTE**, **BASE64**, **DECIMAL, DEFAULT-FOR-EMPTY**, **EMBED-VALUES**, **TEXT**, **USE-NIL**, **USE-NUMBER**, **USE-ORDER**, **USE-QNAME**, **USE-TYPE**, **USE-UNION**, **WHITESPACE**.

18.2.14 There shall be no qualifying information in the "TargetList".

18.3 Effect on encodings

18.3.1 If the type is encoded as a top-level type, this encoding instruction shall be ignored.

18.3.2 The "ExtendedXMLNamedValue" for this component shall not be included in the "ExtendedXMLSequenceValue" or "ExtendedXMLSetValue" of the enclosing sequence or set type. Instead, the value of the enclosing type shall be encoded using the value of each **UTF8String** as an "Attribute" (see clause 20) of the enclosing element as specified below.

18.3.3 The encoder shall:

a) treat each "URI" that is present in a **UTF8String** as requiring that the following "NCName" (the attribute name) be namespace-qualified with the namespace specified by the "URI", and treat the absence of a "URI" in a **UTF8String** as specifying that the following "NCName" shall not be namespace-qualified, and shall then remove the "URI" from the **UTF8String**; and

b) insert into the encoding any necessary namespace declarations with scopes that include the inserted attributes, in order to ensure that the required namespace-qualification of the "NCName"s identified in a) above can be achieved; and

c) insert each **UTF8String** (after the "URI" has been removed) as an attribute in the enclosing element, inserting namespace prefixes as necessary before each "NCName" in order to ensure that the requirements of a) above are satisfied.

18.3.4 The order of all attributes in the enclosing element (resulting from the presence of one or more components of the enclosing type with a final **ATTRIBUTE** or **ANY-ATTRIBUTES** encoding instruction) is an encoder's option.

18.3.5 An EXTENDED-XER decoder shall generate a **UTF8String** in the format of 18.2.6 for each attribute in the enclosing element that is not from the control namespace, and whose name is not that of the identifier (possibly modified in accordance with any final **NAME** or **NAMESPACE** encoding instructions) of another component of the enclosing type that has a final **ATTRIBUTE** encoding instruction.

19 The ANY-ELEMENT encoding instruction

19.1 General

19.1.1 The "AnyElementInstruction" is:

AnyElementInstruction ::=
ANY-ELEMENT
TargetList
NamespaceRestriction ?

19.1.2 The "TargetList" production is defined in 14.2.

19.1.3 The "NamespaceRestriction" is defined in 18.1.

19.1.4 This encoding instruction enables an ASN.1 type that is a **UTF8String** to provide the specification of a single XML element.

NOTE – The content and attributes of the XML element are unrestricted. It may have attributes or child elements, and names of child elements and attributes may be qualified or unqualified, and are not affected by any "NamespaceRestriction".

19.1.5 If there is a "NamespaceRestriction", then the element name is required to satisfy the "NamespaceRestriction" (see 18.1.6 to 18.1.8) but is otherwise unrestricted.

19.1.6 The **UTF8String** with this final encoding instruction may be the root type of the encoding, or may be a component of a choice, sequence, set, sequence-of or set-of type. If it is a top-level type, the type reference name is ignored. If it is a component, the component name is ignored.

19.2 Restrictions

19.2.1 An ASN.1 type shall not have this final encoding instruction unless it is a **UTF8String** type. The component is required to have a restriction applied to it that imposes the format and content specified in 19.2.4 to 19.2.9 by reference to this clause 19 or otherwise.

NOTE – It is recommended that the constraint on the **UTF8String** be expressed as:

```
(CONSTRAINED BY
  {/* Shall conform to the "AnyElementFormat" specified in
      ITU-T Rec. X.693 | ISO/IEC 8825-4, clause 19. */})
```

19.2.2 There shall be no final **UNTAGGED** encoding instruction on the type.

19.2.3 Each "URIList" shall contain at most one occurrence of **ABSENT** and shall not contain two identical "QuotedURI"s.

19.2.4 The format of the abstract values of the **UTF8String** shall conform to the production "AnyElementFormat":

AnyElementFormat ::=
xmlcstring

19.2.5 The "xmlcstring" shall be a syntactically correct XML element defined in W3C XML 1.0 and W3C XML Namespaces.

19.2.6 It shall use only namespace prefixes that are declared in namespace declarations present in the "xmlcstring". If there are unprefixed qualified names, a corresponding default namespace declaration shall be present.

19.2.7 The value of the **UTF8String** shall not cause 10.2.11 to be violated.

19.2.8 If there is a "NamespaceRestriction" of **FROM**, then the (outermost) element name in "AnyElementFormat" shall be the "URI" in a "QuotedURI" in the "URIList", and may be absent only if the keyword **ABSENT** occurs in the "URIList".

19.2.9 If there is a "NamespaceRestriction" of **EXCEPT**, then the (outermost) element name in "AnyElementFormat" shall not be the "URI" in a "QuotedURI" in the "URIList", and shall not be absent if the keyword **ABSENT** occurs in the "URIList".

19.2.10 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

19.2.11 A type with this final encoding instruction shall not also have any of the final encoding instructions **ATTRIBUTE**, **BASE64**, **DEFAULT-FOR-EMPTY**, **PI-OR-COMMENT**, **UNTAGGED** or **WHITESPACE**.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **DECIMAL**, **EMBED-VALUES**, **LIST**, **TEXT**, **USE-NIL**, **USE-NUMBER**, **USE-ORDER**, **USE-QNAME**, **USE-TYPE**, **USE-UNION**.

19.2.12 There shall be no qualifying information in the "TargetList".

19.3 Effect on encodings

19.3.1 An EXTENDED-XER encoder shall include the abstract value of the **UTF8String** in the encoding as an XML element in place of an XML element that would otherwise be generated for this component (ignoring the identifier of the component), or for the root type. The element included shall be identical to the abstract value of the **UTF8String,** except as specified in 19.3.2.

19.3.2 Any namespace declarations that are present in the first start-tag (or empty-element tag) of the element and are identical to namespace declarations that are in-scope at the point of insertion may (but need not) be removed, as an encoder's option.

NOTE – Changing, moving, or deleting other namespace declarations in the **UTF8String** has not been allowed, as such actions may affect the namespace and qualification of XML QNames present in element content or attribute values, and it is generally not possible for an encoder to determine whether such content or attribute values are QNames or not.

19.3.3 An EXTENDED-XER decoder shall generate the format of 19.2.4 from the incoming XML document, as the abstract value of the **UTF8String**.

19.3.4 The decoder shall include, in the first start-tag (or empty-element tag) in the abstract value of the **UTF8String,** namespace declaration attributes for all namespace declarations that are in scope for the element being decoded but that are not present in the start-tag of that element.

20 The ATTRIBUTE encoding instruction

20.1 General

20.1.1 The "AttributeInstruction" is:

AttributeInstruction ::=
ATTRIBUTE
TargetList

20.1.2 The "TargetList" production is defined in 14.2.

20.1.3 This encoding instruction specifies that a character-encodable ASN.1 type is to be encoded as an XML attribute.

NOTE – A particular (but important) case of a character-encodable type is a choice type (all of whose alternatives are character-encodable types) that has a final **USE-UNION** encoding instruction.

20.2 Restrictions

20.2.1 An ASN.1 type shall not have this final encoding instruction unless it has at least one "ExtendedXMLValue" encoding (taking account of encoder's options), for each of its abstract values, that does not contain any XML tags and does not rely on the use of "xmlhstring" (if the type is an open type or octetstring type) or "xmlbstring" (if the type is a bitstring type) or on a final **UNTAGGED**, **ATTRIBUTE**, or **ANY-ATTRIBUTES** encoding instruction applied to its components (if the type is a sequence or set type) to achieve this.

NOTE 1 – This implies that a restricted character string type with a final **ATTRIBUTE** encoding instruction has to be restricted so that it does not contain any of the control characters listed in ITU-T Rec. X.680 | ISO/IEC 8824-1, Table 3 (Escape sequences for control characters in an "xmlcstring"), or has to have a final **BASE64** encoding instruction.

NOTE 2 – This does not include open types, or octetstring and bitstring types with **CONTAINING** without **ENCODED BY**, because their "ExtendedXMLValue" can contain tags unless they are encoded as an "xmlhstring".

NOTE 3 – It is recognized that some ASN.1 tools may not be able to statically check that the above restriction will be satisfied for all abstract values, but conforming encoders cannot generate encodings in which the "ExtendedXMLValue" violates this restriction (see 20.3.14).

20.2.2 A type with this final encoding instruction shall only be used as a component of a sequence or set type.

NOTE – The component may be **OPTIONAL** or **DEFAULT**.

20.2.3 There shall be no final **UNTAGGED** encoding instruction on either the type that has this final encoding instruction or on the enclosing type that contains it as a component.

20.2.4 If the final encoding instructions on other components of the enclosing type include either this encoding instruction or the **ANY-ATTRIBUTES** encoding instruction, 20.3.11 shall not be violated.

20.2.5 A type with this final encoding instruction shall not also have any of the final encoding instructions **ANY-ELEMENT**, **DEFAULT-FOR-EMPTY**, **PI-OR-COMMENT** or **UNTAGGED**.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **EMBED-VALUES**, **USE-NIL**, **USE-ORDER**, **USE-TYPE**.

20.2.6 There shall be no qualifying information in the "TargetList".

20.3 Effect on encodings

20.3.1 If the type is encoded as a top-level type, this encoding instruction shall be ignored.

20.3.2 The "ExtendedXMLNamedValue" of this component shall not be included in the "ExtendedXMLSequenceValue" or "ExtendedXMLSetValue" of the enclosing sequence or set type. Instead, the value of the component (if present) shall be encoded as an "Attribute" (see 20.3.3 to 20.3.15) of the enclosing element.

20.3.3 The "Attribute" production is:

Attribute ::=
AttributeName
"="
QuotedValue

AttributeName ::=
IdentifierOrModifiedIdentifier
| ControlAttributeName

QuotedValue ::=
DoubleQuotedValue
| SingleQuotedValue

DoubleQuotedValue ::=
""" & CharacterEncodableValue & """

SingleQuotedValue ::=
"'" & CharacterEncodableValue & "'"

ControlAttributeName ::= QualifiedName

CharacterEncodableValue ::= ExtendedXMLValue

20.3.4 The "IdentifierOrModifiedIdentifier" production is defined in 17.5.1, and its use in the context of this encoding instruction is defined in 17.6.3.

20.3.5 The "ControlAttributeName" production is not directly used by this clause. All "QualifiedName"s in this production are from the control namespace (see 16.9). Such attributes are only generated in accordance with clauses 33 and 37, but unexpected control attributes are required to be accepted by decoders (see 10.2.10).

20.3.6 The "QualifiedName" is defined in 29.3.2.

20.3.7 The "ExtendedXMLValue" is defined in 17.4.

20.3.8 The "AttributeName" shall be either the "identifier" of the component that has this final encoding instruction or, if there are final **`NAME`** or **`NAMESPACE`** encoding instructions, the "QualifiedOrUnqualifiedName" determined by those encoding instructions as specified in clauses 28 and 29.

20.3.9 The "CharacterEncodableValue" in the "QuotedValue" of the attribute (see 20.3.3) shall be the "ExtendedXMLValue" of this type, possibly modified as specified in 20.3.12 to 20.3.15.

20.3.10 The order in which "Attribute"s appear in an "AttributeList" is an encoder's option, whether these are generated by this encoding instruction or by the **`ANY-ATTRIBUTES`** encoding instruction.

NOTE – No semantics can be placed on the order of attributes in any EXTENDED-XER encoding. This restriction is required by W3C XML 1.0, 3.1.

20.3.11 When an "AttributeList" in an instance of an encoding contains multiple attributes, then for any two "Attribute"s in the list:

a) if the "AttributeName"s of the two attributes are both unqualified names, then they shall be different;

b) if the "AttributeName"s of the two attributes are both namespace-qualified names, then they shall either have different namespace names, or shall be different names in the same namespace.

It is an illegal use of encoding instructions if this condition is violated by the application of final encoding instructions for any abstract value of the top-level type that is being encoded.

20.3.12 If the "QuotedValue" is a "DoubleQuotedValue", and the "ExtendedXMLValue" in the "CharacterEncodableValue" contains a QUOTATION MARK (34) character, then that character shall be replaced by the characters:

`&quot;`

or, as an encoder's option, by an escape sequence of the form **`&#n;`** or **`&#xn;`**, specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.8.

20.3.13 If the "QuotedValue" is a "SingleQuotedValue" and the "ExtendedXMLValue" in the "CharacterEncodableValue" contains an APOSTROPHE (39) character, then that character shall be replaced by the characters:

'

or, as an encoder's option, by an escape sequence of the form **&#n;** or **&#xn;**, specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.8.

20.3.14 The "ExtendedXMLValue" in the "CharacterEncodableValue" shall be one of the encodings of the character-encodable type that does not contain XML tags.

20.3.15 If the "ExtendedXMLValue" contains HORIZONTAL TABULATION (9), LINE FEED (10), or CARRIAGE RETURN (13) characters, then these characters shall be replaced in the "ExtendedXMLValue" by escape sequences of the form "**&#n;**" or "**&#xn;**" specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.8.

21 The BASE64 encoding instruction

21.1 General

21.1.1 The "Base64Instruction" is:

Base64Instruction ::=
BASE64
TargetList

21.1.2 The "TargetList" production is defined in 14.2.

21.1.3 This encoding instruction can be assigned to an **OCTET STRING**, to an open type or to any restricted character string type.

21.1.4 Application of this final encoding instruction to an octet string type or an open type removes the option of a hexadecimal encoding, but allows the option of a Base64 encoding (as specified in IETF RFC 2045, 6.8). Application of this final encoding instruction to a restricted character string type requires that the value of the restricted character string type be encoded as a Base64 encoding.

21.2 Restrictions

21.2.1 If the final encoding instructions for an ASN.1 type contain a **BASE64** encoding instruction then the type shall be:

a) an **OCTET STRING**; or

b) an open type; or

c) a restricted character string type.

21.2.2 A type with this final encoding instruction shall not have any of the final encoding instructions **ANY-ELEMENT** or **WHITESPACE**.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **DECIMAL**, **EMBED-VALUES**, **LIST**, **TEXT**, **USE-NIL**, **USE-NUMBER**, **USE-ORDER**, **USE-QNAME**, **USE-TYPE**, **USE-UNION**.

21.2.3 There shall be no qualifying information in the "TargetList".

21.3 Effect on encodings

21.3.1 This encoding instruction affects only the "ExtendedXMLValue" of the type to which it is applied. It requires the use of either the first or the second alternative of "ExtendedXMLOctetStringValue" and "ExtendedXMLOpenTypeFieldVal" (as an encoder's option), forbidding the third alternative (see 17.4). It requires use of the second alternative of "ExtendedXMLRestrictedCharacterStringValue" (see 17.4).

21.3.2 The "Base64XMLOctetStringValue" is:

Base64XMLOctetStringValue ::=
XMLBase64String

The "XMLBase64String" is defined in 21.3.6.

21.3.3 ITU-T Rec. X.680 | ISO/IEC 8824-1, 22.4, applies.

21.3.4 The "Base64XMLOpenTypeFieldVal" is:

Base64XMLOpenTypeFieldVal ::=
XMLBase64String

21.3.5 The "Base64XMLRestrictedCharacterStringValue" is:

Base64XMLRestrictedCharacterStringValue ::=
XMLBase64String

21.3.6 The "XMLBase64String" is:

XMLBase64String ::=
XMLRestrictedCharacterStringValue

The "XMLRestrictedCharacterStringValue" shall be the Content-Transfer-Encoding specified in IETF RFC 2045, 6.8, except that the 76-character limit does not apply, and "white-space with escapes" (see 8.1.5) is allowed in any position within the "XMLBase64String".

NOTE – IETF RFC 2045 mandates the presence of line breaks dividing the encoding into lines of at most 76 characters, but this is not required in EXTENDED-XER encodings. It also allows "white-space" to be inserted in any position within the base64 encoding.

21.3.7 If applied to a restricted character string type, then each character in the character string shall be encoded with UTF-8 (see ISO 10646, Annex D). The resulting octets for the entire character string shall then be encoded into characters as specified in IETF RFC 2045, 6.8, and the resulting characters shall form the "ExtendedXMLValue".

22 The `DECIMAL` encoding instruction

22.1 General

22.1.1 The "DecimalInstruction" is:

DecimalInstruction ::=
`DECIMAL`
TargetList

22.1.2 The "TargetList" production is defined in 14.2.

22.1.3 The purpose of this encoding instruction is to modify the encoding of a real type so that the exponential notation is forbidden and a hyphen followed by "0" denotes the value plus zero instead of the value minus zero.

NOTE – The value minus zero cannot be represented.

22.2 Restrictions

22.2.1 This encoding instruction shall only be assigned to a real type.

22.2.2 The real type to which this encoding instruction is applied shall be restricted in such a way that the values minus zero, **`MINUS-INFINITY`**, **`PLUS-INFINITY`**, and **`NOT-A-NUMBER`** are not permitted and the **`base`** is 10.

NOTE – It is recommended that this be done by applying the following constraints:

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

22.2.3 This encoding instruction shall not be assigned unless there is a **`GLOBAL-DEFAULTS MODIFIED-ENCODINGS`** encoding instruction in the encoding control section.

22.2.4 A type with this final encoding instruction can have any other final encoding instructions permitted for that type.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **`ANY-ATTRIBUTES, ANY-ELEMENT, BASE64, EMBED-VALUES, LIST, TEXT, USE-NIL, USE-NUMBER, USE-ORDER, USE-QNAME, USE-TYPE, USE-UNION, WHITESPACE.`**

22.2.5 There shall be no qualifying information in the "TargetList".

22.3 Effect on encodings

22.3.1 The "modifiedXMLRealNumber" (see 17.9.3) shall not contain an e or E followed by an exponent.

NOTE – All abstract values, including those that are very large or very small real numbers, are therefore encoded as an integer part optionally followed by a decimal point and a fractional part.

22.3.2 The real value plus zero can be encoded, as an encoder's option, as "XMLDecimalMinusZeroRealValue", defined as follows:

XMLDecimalMinusZeroRealValue ::=
"-" & modifiedXMLRealNumber

where the "modifiedXMLRealNumber" is restricted by 22.3.1 and contains no digits except the digit zero.

NOTE – The above cannot be confused with the real value minus zero, because the value minus zero is removed by the mandatory restriction that applies to the real type (see 22.2.2).

23 The DEFAULT-FOR-EMPTY encoding instruction

23.1 General

23.1.1 The "DefaultForEmptyInstruction" is:

DefaultForEmptyInstruction::=
DEFAULT-FOR-EMPTY
TargetList
AS Value

23.1.2 The "TargetList" production is defined in 14.2.

23.1.3 This encoding instruction specifies an abstract value that can be encoded in an EXTENDED-XER encoding (as an encoder's option) as the "empty" alternative of "ExtendedXMLValue" for a type (see 17.4) that is encoded as the sole content of an XML element.

NOTE – This defaulting mechanism supports the presence of an XML element with no content (typically, but not necessarily, encoded as an empty-element tag). It is distinct from the use of ASN.1 **DEFAULT**, which relates to the absence of the "ExtendedXMLNamedValue" of a component of a sequence or set.

23.1.4 The "TargetList" shall not use the keyword **ALL** and shall identify a single target.

23.1.5 There are five distinct cases where this encoding instruction can be used, identified below.

23.1.5.1 The first case is when it is assigned directly to a character-encodable type that is not **UNTAGGED** (see clause 32). If the enclosing element has empty content, then that empty content represents the specified "Value" of the character-encodable type (which is the governor for "Value").

23.1.5.2 The second case is when it is assigned to a (**NOT UNTAGGED**, **NOT EMBED-VALUES** and **NOT USE-NIL**) sequence type that contains an **UNTAGGED** character-encodable component whose encoding forms the sole content (for all abstract values of the sequence type) of the enclosing element of the sequence type. If the enclosing element of the sequence type has empty content, then that empty content represents the specified "Value" of the character-encodable component (which is the governor for "Value").

NOTE – The character-encodable component may be the sole content because it is the only component, or it may be the sole content because all other components have a final **ATTRIBUTE** (see clause 20) or **ANY-ATTRIBUTES** (see clause 18) encoding instruction.

23.1.5.3 The third case is when it is assigned to a (**NOT UNTAGGED** and **NOT USE-NIL**) sequence type with a final **EMBED-VALUES** encoding instruction (see 25.3.1.4). If the enclosing element of the sequence type has empty content, then that empty content represents an abstract value of the sequence type that would otherwise produce content that is solely the specified "Value" of a sole **UTF8String** in the **EMBED-VALUES** sequence-of (**UTF8String** is the governor for "Value").

23.1.5.4 The fourth case is when it is assigned to a (**NOT UNTAGGED**, **NOT EMBED-VALUES**) sequence type with a final **USE-NIL** encoding instruction (see clause 33) whose **OPTIONAL** component is a character-encodable type. If the enclosing element of the sequence type has a nil identification attribute with value **true**, the **DEFAULT-FOR-EMPTY** does not affect the meaning of the encoding. If the enclosing element of the sequence type has a nil identification attribute with value **false** (or has no nil identification attribute), and has empty content, then that empty content represents the specified "Value" of the **OPTIONAL** component (whose type is the governor for "Value").

23.1.5.5 The fifth case is when it is assigned to a (**NOT UNTAGGED**) sequence type with a final **EMBED-VALUES** encoding instruction (see 25.3.1.4) and a final **USE-NIL** encoding instruction (see clause 33) whose **OPTIONAL** component is a sequence type. If the enclosing element of the sequence type has a nil identification attribute with value **true**, the **DEFAULT-FOR-EMPTY** does not affect the meaning of the encoding. If the enclosing element of the sequence type has a nil identification attribute with value **false** (or has no nil identification attribute), and has empty content, then that empty content represents an abstract value of the sequence type that would otherwise produce content that is solely the specified "Value" of a sole **UTF8String** in the **EMBED-VALUES** sequence-of (the **UTF8String** is the governor for "Value").

23.1.6 "Value" is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, 16.7.

NOTE – This permits use of a value reference defined in or imported into the module. The value reference can be defined using XML Value Notation, but such notation cannot be used directly in "DefaultForEmptyInstruction".

23.2 Restrictions

23.2.1 If the final encoding instructions for an ASN.1 type that is a **NOT UNTAGGED** character-encodable type contain a **DEFAULT-FOR-EMPTY** encoding instruction, then that type shall not be a component (of an ASN.1 **SEQUENCE** or **SET**) with an ASN.1 **DEFAULT** value.

NOTE – This restriction is not strictly necessary, but is imposed to avoid confusion between the normal ASN.1 and the EXTENDED-XER defaulting mechanisms.

23.2.2 This encoding instruction shall only be assigned to:

a) a character-encodable type without a final **UNTAGGED** encoding instruction; or

b) a **NOT UNTAGGED** sequence type, without a final **EMBED-VALUES**, or **USE-NIL** encoding instruction, one of whose components is a character-encodable type with a final **UNTAGGED** encoding instruction and all other components (if any) have a final **ATTRIBUTE** or **ANY-ATTRIBUTES** encoding instruction; or

c) a **NOT UNTAGGED** sequence type, without a final **USE-NIL** encoding instruction, but with a final **EMBED-VALUES** encoding instruction (see 25.3.1.4); or

d) a **NOT UNTAGGED** sequence type, without a final **EMBED-VALUES** encoding instruction, but with a final **USE-NIL** encoding instruction, whose **OPTIONAL** component is a character-encodable type; or

e) a **NOT UNTAGGED** sequence type with a final **EMBED-VALUES** encoding instruction and with a final **USE-NIL** encoding instruction, whose **OPTIONAL** component is a sequence type.

23.2.3 If 23.2.2 a) applies, and "empty" is a valid "ExtendedXMLValue" for one of the abstract values (V, say) of the (possibly constrained) type, and V is different from the "Value" in the "DefaultForEmptyInstruction", then there shall be at least one alternative encoding for V.

23.2.4 If 23.2.2 b) or d) applies, and "empty" is a valid "ExtendedXMLValue" for one of the abstract values (V, say) of the **UNTAGGED** component (case b)) or of the **OPTIONAL** component, (case d)), and V is different from the "Value" in the "DefaultForEmptyInstruction", then there shall be at least one alternative encoding for V.

NOTE – It is recognized that some ASN.1 tools may not be able to statically check that the above restrictions will be satisfied for all abstract values, but conforming encoders cannot generate encodings in which the "ExtendedXMLValue" violates this restriction.

23.2.5 If 23.2.2 c) applies, the **SEQUENCE** type shall be constrained so that (without **DEFAULT-FOR-EMPTY**) there is no abstract value that would produce an empty content for the enclosing element.

23.2.6 If a character-encodable type (case 23.2.2 a)) with this final encoding instruction has an enclosing type that is a sequence-of or set-of type with a final **LIST** encoding instruction, or that is a choice type with a **USE-UNION** encoding instruction, then this final encoding instruction shall be ignored.

23.2.7 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

23.2.8 A type with this final encoding instruction shall not have any of the final encoding instructions **ANY-ELEMENT**, **ATTRIBUTE** or **UNTAGGED**.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **USE-TYPE**.

23.2.9 There shall be no qualifying information in the "TargetList".

23.3 Effect on encodings

23.3.1 This encoding instruction affects only the "ExtendedXMLValue" of the type that is the governor of "Value", (see 23.1.5).

23.3.2 The "ExtendedXMLValue" encoding of the abstract value specified by "Value" shall, as an encoder's option, be either:

a) the "ExtendedXMLValue" encoding of that value which would be produced if the **DEFAULT-FOR-EMPTY** was not present (the normal encoding); or

b) "empty".

NOTE – Decoders are required to accept both the normal encoding and the "empty" encoding as a denotation of the default-for-empty value.

23.3.3 If 23.2.2 a) applies, and "empty" is a valid "ExtendedXMLValue" for one of the abstract values (V, say) of the type, and V is different from the "Value" specified in the "DefaultForEmptyInstruction", then any one of the alternative encodings for V shall be used (as an encoder's option) instead of "empty".

23.3.4 If 23.2.2 b) or d) applies, and "empty" is a valid "ExtendedXMLValue" for one of the abstract values (V, say) of the **UNTAGGED** component (case b)) or of the **OPTIONAL** component (case d)), and V is different from the "Value" specified in the "DefaultForEmptyInstruction", then any one of the alternative encodings for V shall be used (as an encoder's option) instead of "empty".

23.3.5 If 23.2.2 c) applies, the effect of this encoding instruction is specified in 25.3.1.4 and 25.3.1.5.

23.3.6 If 23.2.2 e) applies, the effect of this encoding instruction is specified in 25.3.1.6.

24 The ELEMENT encoding instruction

24.1 General

24.1.1 The "ElementInstruction" is:

ElementInstruction ::=
ELEMENT
TargetList

24.1.2 The "TargetList" production is defined in 14.2.

24.1.3 This encoding instruction is synonymous with **NOT UNTAGGED**, and does not imply any semantics other than **NOT UNTAGGED**.

24.2 Restrictions

24.2.1 There shall be no qualifying information in the "TargetList".

24.2.2 This encoding instruction should not be used as a prefixed encoding instruction in combination with any of the prefixed encoding instructions **ANY-ATTRIBUTES**, **ANY-ELEMENT** or **ATTRIBUTE** to avoid confusing the reader.

24.3 Effect on encodings

This encoding instruction negates an **UNTAGGED** encoding instruction, and does not otherwise affect encodings.

25 The EMBED-VALUES encoding instruction

25.1 General

25.1.1 The "EmbedValuesInstruction" is:

EmbedValuesInstruction ::=
EMBED-VALUES
TargetList

25.1.2 The "TargetList" production is defined in 14.2.

25.1.3 This encoding instruction enables the first component of a (**NOT UNTAGGED**) sequence type to provide character strings to be inserted before the first XML element, after the last XML element, and between the XML elements, that form the "ExtendedXMLValue" encoding of the sequence type.

25.1.4 If a final **USE-NIL** encoding instruction is also present, and the **OPTIONAL** component supporting **USE-NIL** is absent in a particular abstract value, then there will be no XML elements for components of the sequence type, and no character string are provided for that abstract value. Otherwise, for all abstract values, the number of character strings provided is required to be equal to one greater than the number of elements in the encoding of the sequence type. Some or all of the character strings may be empty.

25.2 Restrictions

25.2.1 An ASN.1 type shall not have this final encoding instruction unless it is a sequence type. The first component of the sequence shall be a **SEQUENCE OF UTF8String** and shall not be marked **OPTIONAL** or **DEFAULT**.

25.2.2 There shall be no final **UNTAGGED** encoding instruction (see clause 32) on either the sequence-of type or on the component of the sequence-of.

25.2.3 There shall be no final **UNTAGGED** encoding instruction on any component of the sequence type that is a character-encodable type.

25.2.4 If the sequence type also has a final **USE-NIL** encoding instruction, the **OPTIONAL** component supporting the **USE-NIL** encoding instruction shall not be a character-encodable type (see also 33.2.4).

25.2.5 None of the components of the sequence shall be marked **DEFAULT** unless they have a final **ATTRIBUTE** encoding instruction. If there are components of a **SEQUENCE** or **SET** type (at any depth) that, through the use of **UNTAGGED**, can produce elements in the "ExtendedXMLValue" that are immediate child elements of the sequence type, these shall not be marked **DEFAULT**.

25.2.6 The sequence type shall be restricted in such a way that:

a) if the type has also a final **USE-NIL** encoding instruction and the **OPTIONAL** component supporting **USE-NIL** is absent, the number of repetitions of the sequence-of component is required to be zero;

b) otherwise, the number of repetitions of the sequence-of component in every abstract value equals one plus the number of XML elements in the "ExtendedXMLValue" of the sequence type, determined after application of all final encoding instructions to the other components of the sequence, and ignoring the first component.

NOTE – It is recommended that the constraint on the sequence type be expressed as:

```
(CONSTRAINED BY
        {/* Shall conform to ITU-T Rec. X.693 | ISO/IEC 8825-4, clause 25 */})
```

25.2.7 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

25.2.8 A type with this final encoding instruction shall not have a final **UNTAGGED** encoding instruction.

NOTE – The following final encoding instructions can never occur together with this encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **ATTRIBUTE**, **BASE64**, **DECIMAL**, **LIST**, **TEXT**, **USE-NUMBER**, **USE-QNAME**, **USE-TYPE**, **USE-UNION**, **WHITESPACE**.

25.2.9 There shall be no qualifying information in the "TargetList".

25.3 Effect on encodings

25.3.1 An encoder shall first produce a partial "ExtendedXMLValue" encoding of the enclosing sequence type, ignoring the first component. It shall then modify this encoding as specified in the following subclauses.

NOTE – The **UTF8String** values that are being inserted may be "empty".

25.3.1.1 The first **UTF8String** value in the sequence-of shall be inserted (subject to 25.3.1.6) at the beginning of the partial encoding, before the start-tag of the first XML element (if any).

25.3.1.2 Each successive **UTF8String** value (if any) shall be inserted between the end-tag of an XML element and the start-tag of the following XML element, proceeding from the first element to the last element.

NOTE – The above implies that no **UTF8String** value is inserted inside any of these elements, even if they have child elements.

25.3.1.3 The last **UTF8String** value (if there is one) shall be inserted at the end of the partial encoding, after the end-tag of the last XML element.

25.3.1.4 If no XML elements are present in the partial encoding, and there is also a final **DEFAULT-FOR-EMPTY** encoding instruction (see clause 23) on the sequence type, and the value of the first (and only) **UTF8String** in the sequence-of is identical to the "Value" specified in the **DEFAULT-FOR-EMPTY** encoding instruction, an encoder can optionally encode the **UTF8String** as an empty string (but see 25.3.1.6).

25.3.1.5 If no XML elements are present in the partial encoding, and there is also a final **DEFAULT-FOR-EMPTY** encoding instruction on the sequence type, and the encoding is empty, a decoder shall interpret it as an encoding for the "Value" specified in the **DEFAULT-FOR-EMPTY** encoding instruction and assign this abstract value to the first (and only) **UTF8String** in the sequence-of (but see 25.3.1.6).

NOTE – This means that a value with no XML elements present and with a single empty UTF8String value cannot be encoded. The sequence type is required to be constrained to prohibit such values (see 23.2.5).

25.3.1.6 If the type also has a final **USE-NIL** encoding instruction and the **OPTIONAL** component is absent, then the **EMBED-VALUES** encoding instruction has no effect. If the type also has a final **USE-NIL** encoding instruction and the **OPTIONAL** component is present, then 25.3.1.4 applies. If a decoder determines that the **OPTIONAL** component is present, by the absence of a nil identification attribute (or its presence with the value false), then 25.3.1.5 applies.

26 The GLOBAL-DEFAULTS encoding instruction

26.1 General

26.1.1 The "GlobalDefaultsInstruction" is:

GlobalDefaultsInstruction ::=
GLOBAL-DEFAULTS TargetList DefaultSetting

DefaultSetting ::=
ControlNamespace
| MODIFIED-ENCODINGS

ControlNamespace ::=
CONTROL-NAMESPACE
QuotedURI
Prefix ?

26.1.2 The "TargetList" production is defined in 14.2, and shall be "empty".

26.1.3 "QuotedURI" and "Prefix" are defined in 29.1.1.

26.1.4 The "ControlNamespace" production specifies the name of the control namespace (the "URI" in the "QuotedURI"), and a recommended prefix for that namespace. If this **GLOBAL-DEFAULTS** encoding instruction is not present, the control namespace shall be that specified in 16.9.

26.1.5 The use of **MODIFIED-ENCODINGS** produces "ExtendedXMLValues" that are modified in accordance with 10.2.7 and 10.2.8.

26.2 Restrictions

26.2.1 The **GLOBAL-DEFAULTS** encoding instruction shall be assigned only in an encoding control section and shall not be preceded by any other encoding instructions except other **GLOBAL-DEFAULTS** encoding instructions.

26.2.2 Each of the alternatives of **GLOBAL-DEFAULTS** shall be used at most once in any encoding control section.

26.2.3 The **GLOBAL-DEFAULTS MODIFIED-ENCODINGS**, if present, shall be the first encoding instruction in the XER encoding control section in an ASN.1 module.

26.3 Effect on encodings

26.3.1 The application of **MODIFIED-ENCODINGS** requires that encodings shall be modified as specified in 10.2.7 and 10.2.8.

26.3.2 The control namespace used for an entire XML document shall be the control namespace assigned to the ASN.1 type whose encoding forms the root element of that XML document.

27 The LIST encoding instruction

27.1 General

27.1.1 The "ListInstruction" is:

ListInstruction ::=
LIST
TargetList

27.1.2 The "TargetList" production is defined in 14.2.

27.1.3 This encoding instruction requires that the "ExtendedXMLSequenceOfValue" or "ExtendedXMLSetOfValue" of a sequence-of or set-of type (see 17.7) be the "ExtendedXMLListValue", producing a space-separated list for the values of the component of the sequence-of or set-of.

NOTE – A common assignment of this encoding instruction is to a **SEQUENCE OF INTEGER**, to which an **ATTRIBUTE** encoding instruction (see clause 20) is also assigned.

27.2 Restrictions

27.2.1 The type to which this encoding instruction is assigned shall be a sequence-of or a set-of type.

27.2.2 The component of the sequence-of or set-of type:

a) shall be a character-encodable type; and

b) shall be such that, for all of its abstract values, there is at least one "ExtendedXMLValue" encoding (taking account of all encoder's options) that is not "empty" and that does not contain "white-space with escapes" (see 8.1.5).

NOTE 1 – The above restrictions imply that the component cannot itself be a sequence-of or set-of type with a **LIST** encoding instruction, or contain a nested sequence-of or set-of type with a **LIST** encoding instruction at any depth.

NOTE 2 – The above restrictions will be satisfied if the type of the component of the sequence-of or set-of is the integer type, real type, object identifier type, relative object identifier type, or the **GeneralizedTime** and **UTCTime** useful types. They will also be satisfied if it is a character string type constrained so that it always has at least one character in the character string and none of its values contains a "white-space" character.

NOTE 3 – It is recognized that some ASN.1 tools may not be able to statically check that the above rules are satisfied, but a conforming encoder shall not generate encodings that violate b) above.

27.2.3 A type with this final encoding instruction shall not have a final **ANY-ATTRIBUTES** encoding instruction.

NOTE – The following final encoding instructions can never occur together with this encoding instruction because their application to the type is forbidden: **ANY-ELEMENT**, **BASE64**, **DECIMAL**, **EMBED-VALUES**, **TEXT**, **USE-NIL**, **USE-NUMBER**, **USE-ORDER**, **USE-QNAME**, **USE-TYPE**, **USE-UNION**, **WHITESPACE**.

27.2.4 There shall be no qualifying information in the "TargetList".

27.3 Effect on encodings

27.3.1 This encoding instruction affects only the encoding of the type to which it is applied.

27.3.2 The "ExtendedXMLSequenceOfValue" or "ExtendedXMLSetOfValue" production (see 17.7) shall be the "ExtendedXMLListValue" alternative. "ExtendedXMLListValue is:

ExtendedXMLListValue ::=
empty
| CharacterEncodableValueExtendedXMLListValue

27.3.3 There shall be "white-space with escapes" (see 8.1.5) between each pair of adjacent "CharacterEncodableValue"s in the "ExtendedXMLListValue".

27.3.4 The "CharacterEncodableValue" is defined in 20.3.3. Each "CharacterEncodableValue" shall encode a value of a component of the sequence-of or set-of.

27.3.5 The order in which the "CharacterEncodableValue"s appear in the "ExtendedXMLListValue" shall be the same order in which the corresponding "ExtendedXMLValue"s would appear in an "ExtendedXMLSequenceOfValue" or "ExtendedXMLSetOfValue" if a final **LIST** encoding instruction were not present.

27.3.6 The "CharacterEncodableValue"s in the "ExtendedXMLListValue" shall not be "empty" and shall not contain "white-space with escapes" (see 8.1.5).

NOTE – Subclause 27.2.2 b) ensures that this is possible, but 27.3.4 may restrict encoder's options.

28 The NAME encoding instruction

28.1 General

28.1.1 The "NameInstruction" is:

NameInstruction ::=
NAME
TargetList
AS
NewNameOrKeyword

NewNameOrKeyword ::=
NewName
| Keyword

NewName ::=
RestrictedCharacterStringValue

Keyword ::=
CAPITALIZED
| UNCAPITALIZED
| UPPERCASED
| LOWERCASED

28.1.2 The "TargetList" production is defined in 14.2.

28.1.3 This encoding instruction has five separate purposes:

a) to change the associated tag name, the attribute name, or the value of a possible type identification attribute ("NewName" with no "QualifyingInformation" in the "TargetList") of the target; or

b) to change the case (or the case of the initial letter) of the associated tag name, the attribute name, or the value of a possible type identification attribute ("Keyword" with no "QualifyingInformation" in the "TargetList") of the target(s); or

c) to change the element name used in an empty-element tag normally (as specified in ITU-T Rec. X.680 | ISO/IEC 8824-1) derived from a specified identifier used in the type definition ("NewName" with "QualifyingInformation" in the "TargetList" which is not **ALL**) of the target; or

d) to change the case (or the case of the initial letter) of the element name used in an empty-element tag normally derived from a specified identifier used in the type definition ("Keyword" with "QualifyingInformation" in the "TargetList" which is not **ALL**) of the target(s); or

e) to change the case (or the case of the initial letter) of the element names used in the "ExtendedXMLValue" encoding derived from any identifier used in the type definition ("Keyword" with "QualifyingInformation" in the "TargetList" which is **ALL**) of the target(s).

NOTE 1 – "NewName" can be used to change the names used in an EXTENDED-XER encoding derived from identifiers or type references, but is rarely useful if the new name could have been used in the first place as an ASN.1 identifier or type reference. Thus the normal use of the **NAME** encoding instruction is for producing required XML element or attribute names when they would otherwise not be allowed because of ASN.1 rules on the case of the initial letter of identifiers or type reference names, or where the ASN.1 rules for distinct identifiers in sequence, set and choice constructions prevent a desired XML encoding.

NOTE 2 – The use of **ALL IN ALL AS CAPITALIZED** to capitalize all identifiers in a module can be particularly useful to provide a common style using initial upper-case letters.

NOTE 3 – If a **NAME** encoding instruction is assigned using a target identified by an "identifier" or "typereference", this affects the name used in an EXTENDED-XER encoding, but does not affect the name that is used to identify the same target in subsequent XER encoding instructions.

28.1.4 The "RestrictedCharacterStringValue" is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 37.

28.2 Restrictions

28.2.1 "NewName" shall not be used if the "QualifyingInformation" is **ALL**.

28.2.2 The **NAME** encoding instruction with "QualifyingInformation" shall only be assigned to the following type definitions:

a) a boolean type definition; or

b) a bitstring type definition with named bits; or

c) an enumerated type definition; or

d) an integer type definition with named numbers.

28.2.3 The "RestrictedCharacterStringValue" in "NewName" when used in the **NAME** encoding instruction shall be either an "NCName" defined in W3C XML Namespaces, clause 2, production 4, or an empty character string. It shall not be an empty character string unless the **NAME** encoding instruction is applied to an alternative of a choice type with a final **USE-UNION** encoding instruction.

NOTE 1 – It is a requirement in W3C XML Namespaces that an "NCName" does not commence with characters that when uppercased are "XML".

NOTE 2 – The "NewNameOrKeyword" production (and hence the "NewName" production) is also used in clause 31. The above restrictions on "RestrictedCharacterStringValue" do not apply to the use in clause 31.

28.2.4 If there is a **GLOBAL-DEFAULTS** encoding instruction with a **MODIFIED-ENCODINGS** keyword, there shall be no "QualifyingInformation" in the "TargetList".

NOTE – This is because empty-element tags are not used in this case. The **TEXT** encoding instruction can instead be used to change the encoding of the individual values of a type.

28.2.5 This encoding instruction should not be used, as a prefixed encoding instruction in combination with any of the prefixed encoding instructions **ANY-ATTRIBUTES, ANY-ELEMENT or UNTAGGED** to avoid confusing the reader.

28.3 Effect on encodings

28.3.1 If the type to which this encoding instruction is applied has a final **ATTRIBUTE** encoding instruction, the "AttributeName" (which is in this case an "IdentifierOrModifiedIdentifier") of the "Attribute" (see 20.3.3) shall be the "QualifiedOrUnqualifiedName" alternative as specified in 28.3.3 to 28.3.6.

28.3.2 If the type to which this encoding instruction is applied does not have a final **ATTRIBUTE** encoding instruction, then the enclosing element tag name (which is "TagName" – see 17.5.1) shall be the "QualifiedOrUnqualifiedName" alternative as specified in 28.3.3 to 28.3.6.

28.3.3 The "IdentifierOrModifiedIdentifier" and "QualifiedOrUnqualifiedName" alternatives shall be used. The "UnprefixedName" in the "QualifiedOrUnqualifiedName" shall be the "identifier" of the component modified according the "NewNameOrKeyword" as specified below.

28.3.4 If the "NewName" alternative is used, the "UnprefixedName" shall be replaced by the "NewName".

28.3.5 If the "Keyword" alternative is used, the "UnprefixedName" shall be modified as specified in the subclauses of this 28.3.5.

28.3.5.1 If the "Keyword" is **CAPITALIZED**, then if the first character of the "UnprefixedName" is a lower-case letter, that character shall be replaced by the upper-case equivalent, otherwise the name is not affected.

28.3.5.2 If the "Keyword" is **UNCAPITALIZED**, then if the first character of the "UnprefixedName" is an upper-case letter, then that character shall be replaced by the lower-case equivalent, otherwise the name is not affected.

28.3.5.3 If the "Keyword" is **UPPERCASED**, then all characters of the "UnprefixedName" that are lower-case letters shall be replaced by their upper-case equivalent. Other characters are unchanged.

28.3.5.4 If the "Keyword" is **LOWERCASED**, then all characters of the "UnprefixedName" that are upper-case letters shall be replaced by their lower-case equivalent. Other characters are unchanged.

28.3.6 The "QualifiedOrUnqualifiedName" shall be a namespace-qualified name if and only if the "Type" has a final **NAMESPACE** encoding instruction.

29 The NAMESPACE encoding instruction

29.1 General

29.1.1 The "NamespaceInstruction" is:

NamespaceInstruction ::=
NAMESPACE
TargetList
NamespaceSpecification ?

NamespaceSpecification ::=
AS
QuotedURI
Prefix ?

Prefix ::=
PREFIX
QuotedNCName

QuotedURI ::=
""" & URI & """

QuotedNCName ::=
""" & NCName & """

29.1.2 The "TargetList" production is defined in 14.2.

NOTE – The most common use of this encoding instruction is **NAMESPACE ALL**.

29.1.3 This encoding instruction enables a namespace name and recommended namespace prefix to be assigned to the target(s).

29.1.4 The "URI" production is not defined in this Recommendation | International Standard, but consists of characters that identify a Uniform Resource Identifier (URI). The syntax (and semantics) of a URI is defined in IETF RFC 2396, and commences with the name of a URI scheme. For allocations of namespace names with the **NAMESPACE** encoding instruction, any URI scheme can be used.

29.1.5 If the "NamespaceSpecification" is absent, then a default is assigned with the recommended "Prefix" set to the "modulereference" and the "URI" set as follows:

a) the URI scheme (see IETF RFC 2396) shall be **urn**;

b) the URN namespace identifier (see IETF RFC 2141) shall be **oid**;

c) the URN Namespace Specific String (see IETF RFC 2141) shall be the "DefinitiveIdentifier" of the module expressed as an "XMLObjectIdentifierValue" (see IETF RFC 3061).

29.1.6 EXAMPLE: With an object identifier value of **{iso standard 1564 modules(0) basic(1)}** the "URI" would be the character string **"urn:oid:1.0.1564.0.1"**.

29.1.7 The "NCName" production is defined in W3C XML Namespaces, clause 2, production 4, and shall not commence with characters that when uppercased are "**XML**".

NOTE – This is a requirement imposed by W3C XML Namespaces.

29.2 Restrictions

29.2.1 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

29.3 Effect on encodings

29.3.1 A namespace-qualified name can be required for an associated tag name, for an attribute name, or for the value of a type identification attribute. A namespace-qualified name is required if the type generating the name has a final **NAMESPACE** encoding instruction.

29.3.2 The "QualifiedOrUnqualifiedName" is:

QualifiedOrUnqualifiedName ::=
QualifiedName |
UnqualifiedName

QualifiedName ::=
PrefixedName |
UnprefixedName

UnqualifiedName ::=
UnprefixedName

PrefixedName ::=
DeclaredPrefix & ":" & UnprefixedName

UnprefixedName ::= NCName

DeclaredPrefix ::= NCName

29.3.3 The encoding of a namespace-qualified name requires either:

a) the use of the "PrefixedName" alternative for "QualifiedName" with the addition to XML elements of further attributes providing namespace declarations (as specified in W3C XML Namespaces); or

b) the use of the "UnprefixedName" alternative for "QualifiedName" with the addition to XML elements of further attributes providing default namespace declarations (as specified in W3C XML Namespaces).

29.3.4 The choice of these two mechanisms and the XML elements to which the namespace declaration attributes are added are an encoder's option.

NOTE 1 – W3C XML Namespaces specifies that a default namespace declaration has in its scope only the name of the element in which it is declared (and of child element names), but not of attributes on that element or child elements.

NOTE 2 – It is recommended, but not required, that the recommended prefix in the **NAMESPACE** encoding instruction be used.

NOTE 3 – Use of the recommended prefix may be inappropriate if **NAMESPACE** encoding instructions with different namespace names but the same recommended prefix are present in the module.

30 The PI-OR-COMMENT encoding instruction

30.1 General

30.1.1 The "PIOrCommentInstruction" is:

PiOrCommentInstruction ::=
PI-OR-COMMENT
TargetList
AS
RestrictedCharacterStringValue
Position

Position ::=
BEFORE-TAG
| BEFORE-VALUE
| AFTER-VALUE
| AFTER-TAG

30.1.2 The "TargetList" production is defined in 14.2.

30.1.3 This encoding instruction causes specified XML processing instructions and/or comments to be inserted before or after the "ExtendedXMLValue" or before or after the associated tags.

NOTE – Subclause 10.2.5 permits an encoder (as an encoder's option) to insert additional XML processing instructions and XML comments.

30.1.4 The "RestrictedCharacterStringValue" is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, clause 37.

30.2 Restrictions

30.2.1 The value of the "RestrictedCharacterStringValue" shall be the concatenation of one or more character strings each of which conforms to the syntax of an XML Processing Instruction specified in W3C XML 1.0, 2.6, or to the syntax of an XML Comment specified in W3C XML 1.0, 2.5, and defines the processing instructions and/or comments that are to be inserted in the XML document.

30.2.2 An ASN.1 type shall not have both a final **UNTAGGED** encoding instruction and a final **PI-OR-COMMENT** encoding instruction.

30.2.3 A type with this final encoding instruction shall not have any of the final encoding instructions **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **ATTRIBUTE** or **UNTAGGED**.

30.2.4 There shall be no qualifying information in the "TargetList".

30.3 Effect on the encodings

30.3.1 If the "Position" is **BEFORE-TAG**, then the processing instructions and/or comments shall be inserted before the associated start-tag or empty-element tag. If that start-tag or empty-element tag is the start of some enclosing "ExtendedXMLValue", then any processing instructions and/or comments inserted before that "ExtendedXMLvalue" (using **BEFORE-VALUE** on the corresponding type) shall precede these processing instructions and/or comments in the XML document.

30.3.2 If the "Position" is **BEFORE-VALUE**, then the processing instructions and/or comments shall be inserted at the start of the "ExtendedXMLValue". If that "ExtendedXMLValue" starts with a tag that is the associated start-tag of some embedded "ExtendedXMLValue", then any processing instructions and/or comments inserted before that associated start-tag (using **BEFORE-TAG** on the corresponding type) shall follow these processing instructions and/or comments in the XML document.

NOTE – In this case the contents of the associated tags is never empty, and the empty-element tag cannot be used.

30.3.3 If the "Position" is **AFTER-VALUE**, then the processing instructions and/or comments shall be inserted at the end of the "ExtendedXMLValue". If that "ExtendedXMLValue" ends with a tag that is the associated end-tag of some embedded "ExtendedXMLValue", then any processing instructions and/or comments inserted after that associated end-tag (using **AFTER-TAG** on the corresponding type) shall precede these processing instructions and/or comment in the XML document.

NOTE – In this case the contents of the associated tags is never empty, and the empty-element tag cannot be used.

30.3.4 If the "Position" is **AFTER-TAG**, then the processing instructions and/or comments shall be inserted after the associated end-tag or empty-element tag. If that end-tag or empty-element tag is the end of some enclosing "ExtendedXMLValue", then any processing instructions and/or comments shall be inserted after that "ExtendedXMLvalue" (using **AFTER-VALUE** on the corresponding type) shall follow these processing instructions and/or comments in the XML document.

31 The TEXT encoding instruction

31.1 General

31.1.1 The "TextInstruction" is:

TextInstruction ::=
    TEXT
    TargetList
    TextToBeUsed ?

TextToBeUsed ::=
    AS
    NewNameOrKeyword

31.1.2 The "TargetList" production is defined in 14.2.

31.1.3 The purpose of this encoding instruction is:

a) in the absence of **GLOBAL-DEFAULTS MODIFIED-ENCODINGS**, to enable values of boolean types, enumerated types, bitstrings with named bits, and integers with named numbers, to be encoded as character strings instead of empty-element tags;

b) in the presence of **GLOBAL-DEFAULTS MODIFIED-ENCODINGS**, to enable the character strings that are used for the values of boolean types, enumerated types, bitstrings with named bits, and integers with named numbers, to be changed.

31.1.4 The "NewNameOrKeyword" is defined in clause 28. The "NewName" in "NewNameOrKeyword" shall contain at least one character.

31.2 Restrictions

31.2.1 This encoding instruction shall only be assigned to the following types, with qualifying information identifying one or more of the identifiers used in the definition of the type (or **true** or **false** for the boolean type):

a) a boolean type definition; or

b) a bitstring type definition with named bits; or

c) an enumerated type definition; or

d) an integer type definition with named numbers.

31.2.2 The final character strings used for the values of the type to which this encoding instruction is assigned shall be distinct.

31.2.3 "NewName" in "NewNameOrKeyword" shall not be used if the "QualifyingInformation" is **ALL**. Subclause 28.2.3 does not apply to this use of "NewNameOrKeyword".

31.2.4 In the absence of a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction, the set of final **TEXT** encoding instructions for a type shall not produce text encodings for some abstract values and empty element encodings for other abstract values.

NOTE – If there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction, then all encodings are text encodings.

31.2.5 If the **TEXT** encoding instruction is applied to a bitstring type with named bits and "NewName" is used, the "NewName" shall not contain "white-space with escapes" (see 8.1.5) and shall not commence with a "0" (DIGIT ZERO) or a "1" (DIGIT ONE).

31.2.6 A type with this final encoding instruction shall not also have a final **USE-NUMBER** encoding instruction.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **BASE64**, **DECIMAL**, **EMBED-VALUES**, **LIST**, **USE-NIL**, **USE-ORDER**, **USE-QNAME**, **USE-TYPE**, **USE-UNION**, **WHITESPACE**.

31.2.7 The "QualifyingInformation" shall always be present.

31.3 Effect on encodings

31.3.1 One of the following five subclauses (31.3.2 to 31.3.6) applies.

31.3.2 If the type is not a bitstring type with named bits and the "TextToBeUsed" is absent, the "ExtendedXMLValue" encoding of each value referenced by the qualifying information for this instruction shall be a character string containing the characters of the identifier (or shall be **true** or **false** in the case of boolean types). For integer types with named values, either the identifiers or the corresponding numbers shall be used (as an encoder's option).

31.3.3 If the type is a bitstring type with named bits and the "TextToBeUsed" is absent, a character string identical to the identifier of the bit shall represent the bit when it is set. Each abstract value shall be encoded as the concatenation (possibly empty) of these character strings for all the bits that are set, separated by "white-space with escapes" (see 8.1.5).

31.3.4 If the type is not a bitstring type with named bits and the "TextToBeUsed" is present, then the following subclauses apply (but see 31.3.5).

31.3.4.1 If the "NewName" alternative is used, the character string used to encode the value identified by the "QualifyingInformation" is "NewName". Each occurrence of the characters "**<**", "**>**", and "**&**" in the "NewName" shall be replaced either by one of the escape sequences "**<**", "**&gt**;", and "**&**" respectively, or by an escape sequence of the form "**&#n;**" or "**&#xn;**", specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, 11.15.8.

31.3.4.2 If the "Keyword" alternative is used, the character string used to encode values of the type is the identifier name, modified as specified below.

31.3.4.3 If the "Keyword" is **CAPITALIZED**, then the first character of the name is replaced by the upper-case equivalent, otherwise the name is not affected.

31.3.4.4 If the "Keyword" is **UNCAPITALIZED**, then the name is unchanged.

31.3.4.5 If the "Keyword" is **UPPERCASED**, then all characters of the name that are lower-case letters are replaced by their upper-case equivalent. Other characters are unchanged.

31.3.4.6 If the "Keyword" is **LOWERCASED**, then all characters of the name that are upper-case letters are replaced by their lower-case equivalent. Other characters are unchanged.

31.3.5 If the type is an integer type with named values, the character strings produced by application of subclauses 31.3.4.1 to 31.3.4.6 shall be used in place of the identifiers. Either the character strings or the corresponding numbers shall be used (as an encoder's option).

31.3.6 If the type is a bitstring type with named bits and the "TextToBeUsed" is present, subclauses 31.3.4.1 to 31.3.4.6 shall be applied to each bit identifier to produce the character string that represents the bit when it is set. The bitstring value shall then be encoded as the concatenation (possibly empty) of these character strings for all the bits that are set, separated by "white-space with escapes".

32 The UNTAGGED encoding instruction

32.1 General

32.1.1 The "UntaggedInstruction" is:

UntaggedInstruction ::=
UNTAGGED
TargetList

32.1.2 The "TargetList" production is defined in 14.2.

32.1.3 (Tutorial) An informal description of the effect of **UNTAGGED** on ASN.1 constructors is provided in Annex B. This clause and its subclauses provide a tutorial introduction illustrating some of the effects of using **UNTAGGED**.

32.1.4 Used (possibly repeatedly and nested) in conjunction with sequence, set, choice, sequence-of, and set-of, it enables an almost arbitrary pattern of XML elements to be specified. Its effect is to remove the XML start-tag that precedes the "ExtendedXMLValue" of the "Type" to which it is applied and the XML end-tag that follows it, resulting in the XML elements normally contained between those tags becoming partial XML content.

32.1.5 Applied to a choice type as a component of a sequence or set, it specifies the inclusion at that point in the sequence (set) of exactly one of the alternatives of the choice type (or none if the choice type is an **OPTIONAL** component). The identifier of the choice type does not appear in the encoding. Some alternatives of the choice type may be XML elements, but others may be partial XML content containing an almost arbitrary pattern of multiple elements, through use of **UNTAGGED** in the definition of those alternatives.

32.1.6 Applied to a sequence-of type as a component of a sequence or set, it specifies the inclusion at that point in the sequence (set) of a specified or arbitrary number of repetitions of the sequence-of component (which may produce a single XML element, or may produce partial XML content if it is itself **UNTAGGED**).

32.1.7 Applied to a sequence (or set) type or a sequence-of (or set-of) type as the alternative of a choice type, it enables that alternative to consist of the partial XML content which is the "ExtendedXMLValue" of the sequence, set, sequence-of or set-of.

32.1.8 A separate function of **UNTAGGED** when applied to a character-encodable type is to enable character content to appear in the encoding of a sequence, with no tags around that content. This use is restricted to a component of a sequence which is not itself untagged.

NOTE – The restriction is in order to simplify the rules needed to ensure easy and unambiguous decoding.

32.2 Restrictions

32.2.1 In all instances of use, the enclosing type shall be a sequence, set, choice, sequence-of, or set-of type.

32.2.2 If the type is a character-encodable type, the enclosing type shall be a sequence type without a final **UNTAGGED** encoding instruction. The type shall not be marked **OPTIONAL** or **DEFAULT**. All the other components of the enclosing sequence type (if any) shall have a final **ATTRIBUTE** or **ANY-ATTRIBUTES** encoding instruction.

32.2.3 If the type is not a character-encodable type, it shall be a sequence, a set, a choice, a sequence-of, a set-of, an octetstring or bitstring type with a contained "Type" without **ENCODED BY**, or an open type.

NOTE – Annex B provides guidelines that can ensure that ambiguities do not result from the use of this encoding instruction.

32.2.4 This encoding instruction shall not be applied to a type that has an empty "ExtendedXMLValue" encoding for one of its abstract values, if the type is used as:

a) a component of a sequence or set type with **OPTIONAL** or **DEFAULT**; or

b) the component of a sequence-of or set-of type; or

c) an alternative of a choice type, if another alternative of the same choice type has an empty "ExtendedXMLValue" encoding for one of its abstract values and has a final **UNTAGGED** encoding instruction.

EXAMPLE: A type that is a sequence type with all of its components **OPTIONAL** has an abstract value with an empty "ExtendedXMLValue" encoding, as does a sequence-of type where zero repetitions are allowed.

32.2.5 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

32.2.6 A type with this final encoding instruction shall not have any of the final encoding instructions **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **ATTRIBUTE**, **DEFAULT-FOR-EMPTY**, **EMBED-VALUES**, **PI-OR-COMMENT**, **USE-NIL**, **USE-ORDER** or **USE-TYPE**.

32.2.7 There shall be no qualifying information in the "TargetList".

32.3 Effect on encodings

32.3.1 If the type is encoded as a top-level type, this encoding instruction shall be ignored.

32.3.2 If the enclosing type is a choice type, the "ExtendedXMLChoiceValue" (see 17.5.1) for this alternative of the enclosed type shall be the "ExtendedXMLValue" of the alternative (the second alternative in the "ExtendedXMLChoiceValue" production).

NOTE – This "ExtendedXMLValue" for the alternative may be a single XML element or may be partial XML content consisting of multiple XML elements.

32.3.3 If the enclosing type is a sequence or set type, the "ExtendedXMLNamedValue" (see 17.6) for this component of the enclosed type shall be replaced by the "ExtendedXMLValue" of the component (the second alternative in the "ExtendedXMLNamedValue" production).

NOTE – This "ExtendedXMLValue" may be a single XML element or may be partial XML content consisting of multiple XML elements.

32.3.4 If the enclosing type is a sequence-of or set-of type, the "ExtendedXMLDelimitedItem" (if used – see 17.7) of each repetition shall be replaced by the "ExtendedXMLValue" enclosed in the "ExtendedXMLDelimitedItem".

NOTE 1 – It is not possible to use **UNTAGGED** unless a **GLOBAL-DEFAULTS** of **MODIFIED-ENCODINGS** has been included in the encoding control section, in which case "ExtendedXMLValueList" is not permitted (see 17.7.2).

NOTE 2 – This "ExtendedXMLValue" may be a single XML element or may be partial XML content consisting of multiple XML elements.

32.3.5 If the type is an octetstring or bitstring type with a contained "Type" without **ENCODED BY**, or an open type, the "ExtendedXMLValue" shall be an "ExtendedXMLTypedValue" (not an "xmlhstring" or an "XMLBase64String").

NOTE – Such types do not match the definition of character-encodable type (see 3.2.2 *ter*). Subclause 32.3.5 implies that when they have a final **UNTAGGED** encoding instruction, they are always encoded as XML elements.

33 The USE-NIL encoding instruction

33.1 General

33.1.1 The "UseNilInstruction" is:

UseNilInstruction ::=
USE-NIL
TargetList

33.1.2 The "TargetList" production is defined in 14.2.

33.1.3 This encoding instruction provides an optimized EXTENDED-XER encoding for a sequence with a single **OPTIONAL** component whose other components (if any) all have a final **ATTRIBUTE** or **ANY-ATTRIBUTES** encoding instruction, possibly preceded by an initial sequence-of type supporting **USE-ORDER** (see clause 35).

33.1.4 In the absence of this encoding instruction, the optional component would encode as follows:

a) (the "not missing but empty" case) if the component is present in the abstract value, with the abstract value that has an empty "ExtendedXMLValue" encoding, an "ExtendedXMLNamedValue" for the component is present in the XML document, usually as an empty-element tag (or with adjacent start and end tags);

b) (the "missing" case) if the component is absent in the abstract value, the "ExtendedXMLNamedValue" is not present;

c) (the "not missing and not empty" case) if the component is present in the abstract value with an abstract value that does not have an empty encoding, an "ExtendedXMLNamedValue" for the component is present with non-empty content.

33.1.5 Use of **USE-NIL** requires that the absence of the optional component (case b) above) be signalled by the inclusion of a nil identification attribute with name "**nil**" and a value of either "**true**" or "**1**".

33.1.6 In cases a) and c) of 33.1.4, the nil identification attribute can either be omitted (as an encoder's option), or it can be present with a value of either "**false**" or "**0**". The optional component shall be encoded by omitting the associated tags.

33.2 Restrictions

33.2.1 The **USE-NIL** encoding instruction shall only be assigned to a sequence type that has an **OPTIONAL** component without a final **ATTRIBUTE** encoding instruction. All the other components of the sequence type, if any, shall have a final **ATTRIBUTE** or **ANY-ATTRIBUTES** encoding instruction, or shall be the sequence-of components supporting a **USE-ORDER** or an **EMBED-VALUES** encoding instruction that are also final encoding instructions on the sequence type.

33.2.2 The sequence type shall not have a final **UNTAGGED** encoding instruction.

33.2.3 The **OPTIONAL** component shall not have any of the final encoding instructions **ANY-ELEMENT**, **ANY-ATTRIBUTES**, **DEFAULT-FOR-EMPTY**, **EMBED-VALUES**, **PI-OR-COMMENT**, **UNTAGGED**, **USE-NIL**, **USE-ORDER** or **USE-TYPE**.

NOTE – Apart from **UNTAGGED**, the encoding instructions listed above are those that cannot be applied to a type that has a final **UNTAGGED** encoding instruction.

33.2.4 If the **OPTIONAL** component is not a character-encodable type, then it shall be a sequence, set, choice, sequence-of, set-of type, an open type, or an octetstring or bitstring type with a contained "Type" and without **ENCODED BY**.

33.2.5 If the **OPTIONAL** component is a sequence type, none of its components shall have a final **ATTRIBUTE** or **ANY-ATTRIBUTES** encoding instruction.

33.2.6 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

33.2.7 A type with this final encoding instruction shall not also have any of the final encoding instructions **UNTAGGED** or **USE-QNAME**.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **ATTRIBUTE**, **BASE64**, **DECIMAL**, **LIST**, **TEXT**, **USE-NUMBER**, **USE-TYPE**, **USE-UNION**, **WHITESPACE**.

33.2.8 There shall be no qualifying information in the "TargetList".

33.3 Effect on encodings

33.3.1 If the **OPTIONAL** component is absent (case b) of 33.1.4), then a nil identification attribute with name "**nil**" and a value of either "**true**" or "**1**" shall be added to the "AttributeList" of the enclosing element.

33.3.2 If the **OPTIONAL** component is present (cases a) and c) of 33.1.4), the nil identification attribute can either be omitted (as an encoder's option), or it can be added to the "AttributeList" of the enclosing element with a value of either "**false**" or "**0**". The optional component shall be encoded by omitting the associated tags.

34 The USE-NUMBER encoding instruction

34.1 General

34.1.1 The "UseNumberInstruction" is:

UseNumberInstruction ::=
USE-NUMBER
TargetList

34.1.2 The "TargetList" production is defined in 14.2.

34.1.3 The purpose of this encoding instruction is to modify the encoding of an enumerated type so that the numbers in the "NamedNumber" enumerations are used instead of the names.

34.2 Restrictions

34.2.1 This encoding instruction shall be ignored unless it is applied to an enumerated type.

34.2.2 A type with this final encoding instruction shall not also have a final **TEXT** encoding instruction.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **BASE64**, **DECIMAL**, **EMBED-VALUES**, **LIST**, **USE-NIL**, **USE-ORDER**, **USE-QNAME**, **USE-TYPE**, **USE-UNION**, **WHITESPACE**.

34.2.3 There shall be no qualifying information in the "TargetList".

34.3 Effect on encodings

34.3.1 The "ExtendedXMLEnumeratedValue" is:

ExtendedXMLEnumeratedValue ::=
EmptyElementEnumerated
| TextEnumerated
| XMLSignedNumber

34.3.2 The "EmptyElementEnumerated" and "TextEnumerated" are defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, 19.8 and 19.9.

34.3.3 The "XMLSignedNumber" is defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, 18.9 and 18.12, and shall be the number in the "NamedNumber" of the enumeration.

34.3.4 The XMLSignedNumber alternative shall be used if and only if the enumerated type has this final encoding instruction.

NOTE – If a **GLOBAL-DEFAULTS** of **MODIFIED-ENCODINGS** is present in the XER Encoding Control Section but the enumerated type does not have this final encoding instruction, then the second alternative is used. If there is no **GLOBAL-DEFAULTS** of **MODIFIED-ENCODINGS** present in the XER Encoding Control Section, then the first alternative is used.

35 The USE-ORDER encoding instruction

35.1 General

35.1.1 The "UseOrderInstruction" is:

UseOrderInstruction ::=
USE-ORDER
TargetList

35.1.2 The "TargetList" production is defined in 14.2.

35.1.3 The purpose of this encoding instruction is to allow an optimized EXTENDED-XER encoding of a sequence type in which there is a sequence-of component that determines the semantic order of the values of the following components of the sequence type that are encoded as elements. It can also be used, if there is also a final **USE-NIL** encoding instruction (see clause 33), and the single **OPTIONAL** component required by the use of **USE-NIL** is a sequence, to determine the semantic order of the components of that **OPTIONAL** sequence.

35.1.4 The sequence-of component that determines the semantic order is the first component of the sequence, unless there is also a sequence-of component supporting a final **EMBED-VALUES** encoding instruction on the sequence type. In this case, the sequence-of component supporting the **EMBED-VALUES** encoding instruction precedes the sequence-of component supporting the **USE-ORDER** encoding instruction.

35.1.5 The component determining the semantic order is required to be a sequence-of type with a component that is an enumerated type. That sequence-of type and its semantics depends on the presence or absence of a **USE-NIL** encoding instruction on the sequence type, as described in the following subclauses.

35.1.5.1 Where there is no final **USE-NIL** encoding instruction, the names of the enumerations are identical to the ASN.1 identifiers of the components of the sequence type. The order of the enumerations in each abstract value determines the semantic order of the values of the following components of the sequence type that are present in the encoding.

35.1.5.2 Where there is also a final **USE-NIL** encoding instruction, the **OPTIONAL** component required by the use of **USE-NIL** is required to be a sequence type (B, say), and the names of the enumerations are identical to the ASN.1 identifiers of the components of the sequence type B. The order of the enumerations in each abstract value determines the semantic order of the values of the components of the sequence type B that are present in the encoding.

35.2 Restrictions

35.2.1 This encoding instruction shall only be assigned to a sequence type. The sequence type shall contain a component that is a sequence-of type (type A, say) with a component that is an enumerated type. If the sequence type does not have also a final **EMBED-VALUES** encoding instruction, then type A shall be the first component, otherwise it shall be the second component. If there is no final **USE-NIL** encoding instruction, the sequence type shall also have at least one other component with no final **ATTRIBUTE** or **ANY-ATTRIBUTES** encoding instruction (a non-attribute component). If there is a final **USE-NIL** encoding instruction, the **OPTIONAL** component supporting the **USE-NIL** shall be a sequence type, and it shall have at least one component.

35.2.2 The enumerated type shall have identifiers that depend on the presence or absence of a final **USE-NIL** encoding instruction on the sequence type with the **USE-ORDER** encoding instruction, as specified in the following subclauses.

35.2.2.1 If there is no final **USE-NIL** encoding instruction, then the enumerated type shall have identifiers for the enumerations that are in one-to-one correspondence (and are in the same textual order) with the identifiers of the following non-attribute components (see 35.2.1) of the sequence. The sequence-of type shall be constrained so that every abstract value contains exactly one identifier for each non-attribute component of the sequence that is present in the abstract value.

35.2.2.2 If there is a final **USE-NIL** encoding instruction, then the enumerated type shall have identifiers for the enumerations that are in one-to-one correspondence (and are in the same textual order) with the identifiers of the components of the **OPTIONAL** component in the sequence type. The sequence-of type shall be constrained so that every abstract value contains exactly one identifier for each component of the **OPTIONAL** sequence that is present in the abstract value.

NOTE – It is recommended that the constraint on the sequence type be expressed as:

```
(CONSTRAINED BY {/* Shall conform to ITU-T Rec. X.693 | ISO/IEC 8825-4, clause 35 */})
```

35.2.2.3 The "EnumerationItem"s in the enumerations shall all be "identifier"s or shall all be "NamedNumber"s with the value 0 for the first "EnumerationItem", 1 for the second, and so on, up to the last "EnumerationItem".

35.2.3 The sequence-of type shall not be marked **OPTIONAL** or **DEFAULT**.

35.2.4 The following components of the sequence (if there is no final **USE-NIL** encoding instruction), and the components of the **OPTIONAL** sequence (if there is a final **USE-NIL** encoding instruction) shall not be marked **DEFAULT** unless they have a final **ATTRIBUTE** encoding instruction.

35.2.5 No component of either the sequence with this final encoding instruction or the **OPTIONAL** sequence (when a final **USE-NIL** encoding instruction is present) shall have a final **UNTAGGED** encoding instruction, whether the type of that component is a character-encodable type or not.

35.2.6 No component of the sequence with this final encoding instruction shall have a final **ANY-ELEMENT** encoding instruction.

35.2.7 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

35.2.8 A type with this final encoding instruction shall not also have a final **UNTAGGED** encoding instruction.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **ATTRIBUTE**, **BASE64**, **DECIMAL**, **LIST**, **TEXT**, **USE-NUMBER**, **USE-QNAME**, **USE-TYPE**, **USE-UNION**, **WHITESPACE**.

35.2.9 There shall be no qualifying information in the "TargetList".

35.3 Effect on encodings

35.3.1 The sequence-of type with the enumerated component shall not be directly encoded.

35.3.2 An encoder shall encode the semantics of this type (the semantic order of the sequence components or of the **OPTIONAL** sequence components) by encoding the components that are encoded as elements in the order specified by the sequence-of type with the enumerated component. A decoder shall recover the value of the sequence-of component by use of the order of the encoded elements.

36 The USE-QNAME encoding instruction

36.1 General

36.1.1 The "UseQNameInstruction" is:

UseQNameInstruction ::=
USE-QNAME
TargetList

36.1.2 The "TargetList" production is defined in 14.2.

36.1.3 The purpose of this encoding instruction is to modify the encoding of a sequence type, each of whose values specifies an optional namespace name (a URI) and an unprefixed name, so that it encodes as an XML namespace-qualified or unqualified name.

NOTE – This is provided because it is available in other schema notations. An example of a sequence type to which it could be applied is the **QName** type defined in ITU-T Rec. X.694 | ISO/IEC 8825-5.

36.1.4 If the optional component is present in an abstract value of the sequence type, then that abstract value represents a namespace-qualified name. If the optional component is absent, the sequence type represents an unqualified name.

36.2 Restrictions

36.2.1 This encoding instruction shall only be assigned to a sequence with exactly two components, both of type **UTF8String**. The first component shall be **OPTIONAL**.

36.2.2 The first component shall be restricted to represent a URI (see IETF RFC 2396). The second component shall be restricted to contain an "NCName" as specified in W3C XML Namespaces, clause 2, production 4, and shall not commence with characters that when uppercased are "**XML**".

36.2.3 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

36.2.4 A type with this final encoding instruction shall not also have a final **USE-NIL** encoding instruction.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **BASE64**, **DECIMAL**, **EMBED-VALUES**, **LIST**, **TEXT**, **USE-NUMBER**, **USE-ORDER**, **USE-TYPE**, **USE-UNION**, **WHITESPACE**.

36.2.5 There shall be no qualifying information in the "TargetList".

36.3 Effect on encodings

36.3.1 The presence of this encoding instruction on a type, if the optional component is present, requires that a namespace declaration (or default namespace declaration) be in scope for the attribute value or element content that encodes the value of this type, in accordance with clause 29. The attribute value or element content is then encoded as specified for a namespace-qualified name in clause 29.

36.3.2 If the optional component is absent, a default namespace declaration shall not be in scope for the attribute value or element content that encodes the value of this type.

37 The USE-TYPE encoding instruction

37.1 General

37.1.1 The "UseTypeInstruction" is:

UseTypeInstruction ::=
USE-TYPE
TargetList

37.1.2 The "TargetList" production is defined in 14.2.

37.1.3 This encoding instruction optimizes the EXTENDED-XER encoding of a choice type. It requires a type identification attribute to be encoded in the enclosing element to identify the alternative that has been encoded (unless this is the first alternative) and the removal of the start-tag and end-tag around the encoding of the alternatives.

37.1.4 The type identification attribute identifies the type of an XML element. The name of the attribute is required to be the name "**type**" from the control namespace (see 16.9) and its value identifies an alternative of the choice type to which this encoding instruction is applied (it provides alternative determination for the choice type).

37.2 Restrictions

37.2.1 The type to which **USE-TYPE** is assigned shall be a choice type without a final **UNTAGGED** encoding instruction.

37.2.2 None of the alternatives of the choice type shall have a final **UNTAGGED** encoding instruction.

37.2.3 None of the alternatives of the choice type shall itself be a choice type with a final **USE-TYPE** encoding instruction.

NOTE – One or more alternatives of the choice type may be choice types with a final **USE-UNION** encoding instruction.

37.2.4 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

37.2.5 A type with this final encoding instruction shall not also have any of the final encoding instructions **UNTAGGED** or **USE-UNION**.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **ATTRIBUTE**, **BASE64**, **DECIMAL**, **DEFAULT-FOR-EMPTY**, **EMBED-VALUES**, **LIST**, **TEXT**, **USE-NIL**, **USE-NUMBER**, **USE-ORDER**, **USE-QNAME**, **WHITESPACE**.

37.2.6 There shall be no qualifying information in the "TargetList".

37.3 Effect on encodings

37.3.1 If the alternative of the choice being encoded is not the first alternative of that choice, then a type identification attribute (see 37.3.3 and 37.3.4) shall be added to the "AttributeList" of the enclosing element, unless 37.3.8 applies.

37.3.2 If the alternative of the choice being encoded is the first alternative, the type identification attribute may be added or omitted as an encoder's option, unless 37.3.8 applies.

37.3.3 The type identification attribute shall be an instance of the "Attribute" production (see 20.3.3) with a namespace-qualified "ControlAttributeName" (see 20.3.5) of "**type**" from the control namespace (see 16.9).

37.3.4 The value of the type identification attribute shall be the identifier of the chosen alternative, possibly modified in accordance with any final **NAME** and **NAMESPACE** encoding instructions.

37.3.5 If there is no type identification attribute present in an encoding of a type with this final encoding instruction, a decoder shall assume that the first alternative of the choice is present.

37.3.6 The presence of a type identification attribute with an unexpected value shall not result in a decoding error. When encountering such an attribute in an encoding, a decoder shall assume that the first alternative of the choice has been identified, and may ignore the type identification attribute (or pass it to the application). In addition, in such cases, the decoder may ignore (or pass to the application) any other unexpected attributes and any unexpected child elements encountered after all the expected child elements in the "ExtendedXMLValue" of the alternative.

37.3.7 All the "Attribute"s that would otherwise be in the "AttributeList" of the "ExtendedXMLChoiceValue" shall be added to the "AttributeList" of the enclosing element and the "ExtendedXMLChoiceValue" of the choice type shall be replaced by the "ExtendedXMLValue" in the "ExtendedXMLChoiceValue".

37.3.8 If one or more alternatives of the choice type with the **USE-TYPE** final encoding instruction are choice types with a final **USE-UNION** encoding instruction, the type identification attribute may, as an encoder's option, identify one of the alternatives of the choice type with the final **USE-UNION** instruction instead of an alternative of the choice type with the **USE-TYPE** final encoding instruction.

38 The USE-UNION encoding instruction

38.1 General

38.1.1 The "UseUnionInstruction" is:

UseUnionInstruction ::=
    USE-UNION
    TargetList

38.1.2 The "TargetList" production is defined in 14.2.

38.1.3 This encoding instruction optimizes the encoding of a choice type in cases where the encoding of the abstract values of each alternative is sufficiently distinct from the encoding of abstract values of other alternatives for a decoder to determine the abstract value represented by analysis of the encoding.

38.1.4 If the choice type with a final **USE-UNION** encoding instruction does not also have a final **ATTRIBUTE** or a final **UNTAGGED** encoding instruction, then this encoding instruction can result in the insertion of a type identification attribute in the enclosing element to identify the alternative that has been encoded. If the choice type has a final **ATTRIBUTE** or **UNTAGGED** encoding instruction, or is the component of a sequence-of or set-of type with a **LIST** encoding instruction, the insertion of the type identification attribute is not possible.

38.1.5 This encoding instruction causes the removal of the start-tag and end-tag around the encoding of the alternative.

38.2 Restrictions

38.2.1 A type with a final encoding instruction of **USE-UNION** shall be a choice type.

38.2.2 All the alternatives of the choice type shall be character-encodable types, but shall not be choice types with a final **USE-UNION** encoding instruction.

38.2.3 If the choice type has a final **ATTRIBUTE** or **UNTAGGED** encoding instruction or is used in a type definition as a component of a sequence-of or set-of type with a final **LIST** encoding instruction, the alternatives of the choice type shall be constrained so that, for any alternative, all its abstract values have at least one encoding (its "ExtendedXMLValue") that is different from all the allowed encodings of all the textually-preceding alternatives.

NOTE – This requirement is imposed because it is impossible to insert a type-identification attribute determining the alternative that was selected. Without this requirement, the encoding would be ambiguous.

38.2.4 In the following two subclauses, the term "identifier" means: identifier (possibly modified in accordance with any final **NAME** and **NAMESPACE** encoding instructions) of an alternative (of the choice type).

38.2.5 If the choice type (type U, say) is being encoded as an alternative of an enclosing choice type (type E, say) that has a final **USE-TYPE** encoding instruction, and the identifier of one of the alternatives of E is identical to the identifier of one of the alternatives of U, then each abstract value of that alternative of U shall have at least one encoding that is different from all the encodings of the textually-preceding alternatives of U.

NOTE – This requirement is imposed because in this case it is not possible to identify the alternative of U, as the identifier in a type identification attribute for E would merely identify the whole of U.

38.2.6 If the choice type (U1, say) is being encoded as an alternative of an enclosing choice type (E, say) with a final **USE-TYPE** encoding instruction, and E contains another choice type (U2, say) with a **USE-UNION** encoding instruction that textually follows U1 in E, and the identifier of any one of the alternatives of U2 is identical to one of the identifiers in U1, then each abstract value of that alternative of U2 shall have at least one encoding that is different from all the encodings of all the alternatives of U1.

NOTE – This requirement is imposed because in this case it is not possible to identify the alternative of U2, as the identifier in a type identification attribute for E would identify the alternative in U1.

38.2.7 This encoding instruction shall not be assigned unless there is a **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** encoding instruction in the encoding control section.

38.2.8 A type with this final encoding instruction shall not also have a final **USE-TYPE** encoding instruction.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **ANY-ELEMENT**, **BASE64**, **DECIMAL**, **EMBED-VALUES**, **LIST**, **TEXT**, **USE-NIL**, **USE-NUMBER**, **USE-ORDER**, **USE-QNAME**, **WHITESPACE**.

38.2.9 There shall be no qualifying information in the "TargetList".

38.3 Effect on encodings

38.3.1 If the choice type does not have a final **ATTRIBUTE** or **UNTAGGED** encoding instruction and is not encoded as the component of a sequence-of or set-of type with a final **LIST** encoding instruction, then a type identification attribute may be added, as an encoder's option, to the "AttributeList" of the enclosing element (but see 38.3.8).

NOTE – If the choice type is encoded as an alternative of a choice with a **USE-TYPE** encoding instruction, the type identification attribute specified by the **USE-UNION** encoding instruction can be used instead of the type identification attribute specified by the **USE-TYPE** encoding instruction (see 37.3.8).

38.3.2 If every possible encoding of the abstract value being encoded is identical to one of the encodings of an abstract value of a textually-preceding alternative, then a type identification attribute shall be added.

NOTE – This subclause 38.3.2 removes the encoder's option of subclause 38.3.1 and makes the addition of the type identification attribute mandatory. The restrictions specified in 38.2.4 to 38.2.6 ensure that this can only occur when the choice type is encoded as an element and when no ambiguity due to identical identifiers is possible.

38.3.3 If the choice type has a final **ATTRIBUTE** or **UNTAGGED** encoding instruction or its enclosing type is a sequence-of or set-of type with a final **LIST** encoding instruction, no type identification attribute can be inserted in any element. In the case of the scenarios described in 38.2.4 to 38.2.6, a type identification attribute cannot be inserted to precisely identify some of the alternatives of U or U2. Decoders shall therefore rely on the conditions of 38.2.4 to 38.2.6 to determine the abstract value that has been encoded.

NOTE – These rules imply that a decoder is required, in the absence of a type identification (or in the presence of an ambiguous one), to attempt to decode against the textually first alternative, then the next, and so on, accepting the first successful decode that is found (or diagnosing an error if there is no successful decode).

38.3.4 The type identification attribute shall be an instance of the "Attribute" production (see 20.3.3) with a namespace-qualified "ControlAttributeName" (see 20.3.5) of "**type**" from the control namespace (see 16.9).

38.3.5 The value of the type identification attribute shall be the identifier of the chosen alternative, possibly modified in accordance with any final **NAME** and **NAMESPACE** encoding instructions.

38.3.6 All the "Attribute"s that would otherwise be in the "AttributeList" of the "ExtendedXMLChoiceValue" shall be added to the "AttributeList" of the enclosing element and the "ExtendedXMLChoiceValue" of the choice type shall be replaced by the "ExtendedXMLValue" in the "ExtendedXMLChoiceValue".

38.3.7 The "ExtendedXMLValue" of the character-encodable type shall be one of the encodings that does not contain any XML tags.

NOTE – This may restrict encoders' options.

38.3.8 If an alternative of the choice type has a final **NAME AS ""** encoding instruction, no type identification attribute shall be added for that alternative.

39 The WHITESPACE encoding instruction

39.1 General

39.1.1 The "WhiteSpaceInstruction" is:

WhiteSpaceInstruction ::=
WHITESPACE
TargetList
WhiteSpaceAction

WhiteSpaceAction ::=
REPLACE
| **COLLAPSE**

39.1.2 The "TargetList" production is defined in 14.2.

39.1.3 This encoding instruction requires decoders to accept additional options in the encoding of the SPACE (32) character and in the use of leading and trailing "white-space with escapes" (see 8.1.5) for character string encodings.

39.2 Restrictions

39.2.1 This encoding instruction can only be assigned to a restricted character string type that either does not contain, or is constrained not to contain the following characters:

a) HORIZONTAL TABULATION (9);

b) LINE FEED (10);

c) CARRIAGE RETURN (13).

39.2.2 If this encoding instruction has the **COLLAPSE** option, then it shall not be applied to a restricted character string type unless that type is constrained not to have leading or trailing spaces or contain multiple adjacent spaces for any abstract value.

NOTE – It is recognized that some ASN.1 tools may not be able to statically check that the above restriction will be satisfied for all abstract values, but conforming encoders cannot generate encodings in which the "ExtendedXMLValue" violates this restriction.

39.2.3 A type with this final encoding instruction shall not also have any of the final encoding instructions **ANY-ELEMENT** or **BASE64**.

NOTE – The following final encoding instructions can never occur together with this final encoding instruction because their application to the type is forbidden: **ANY-ATTRIBUTES**, **DECIMAL**, **EMBED-VALUES**, **LIST**, **TEXT**, **USE-NIL**, **USE-NUMBER**, **USE-ORDER**, **USE-QNAME**, **USE-TYPE**, **USE-UNION**.

39.2.4 There shall be no qualifying information in the "TargetList".

39.3 Effect on encodings

39.3.1 If the keyword **REPLACE** is used, every SPACE (32) in the abstract value can be replaced, as an encoder's option, by a single character that is "white-space with escapes" (see 8.1.5).

39.3.2 If the keyword **COLLAPSE** is used, every SPACE (32) can be replaced, as an encoder's option, by any number of "white-space with escapes" characters. In addition, one or more such characters can be added to the beginning and/or to the end of the "ExtendedXMLValue" encoding as an encoder's option.

Replace the previously existing clause 10 and its subclauses with the following (noting that it has been renumbered as clause 40):

40 Object identifier values referencing the encoding rules

40.1 The encoding rules specified in this Recommendation | International Standard can be referenced and applied whenever there is a need to specify an unambiguous character string representation for the values of a single identified ASN.1 type.

40.2 The following object identifier and object descriptor values are assigned to identify the encoding rules specified in this Recommendation | International Standard:

For BASIC-XER:

```
{joint-iso-itu-t asn1 (1) xml-encoding (5) basic (0)}
"Basic XML encoding of a single ASN.1 type"
```

For CXER:

```
{joint-iso-itu-t asn1 (1) xml-encoding (5) canonical (1)}
"Canonical XML encoding of a single ASN.1 type"
```

For EXTENDED-XER:

```
{joint-iso-itu-t asn1 (1) xml-encoding (5) extended (2)}
"Extended XML encoding of a single ASN.1 type"
```

Insert a new subclause 40.3 as follows:

40.3 The following object identifier and object descriptor values are assigned in order to identify the ASN.1 namespace (see 16.9):

```
asn1Namespace OBJECT IDENTIFIER ::=
            {joint-iso-itu-t asn1 (1) xml-encoding (5) extended (2)
                                        modules (0) support (1) }
            "ASN.1 namespace for EXTENDED-XER support"
```

Replace Annex A with the following:

Annex A

Examples of BASIC-XER and CXER encodings

(This annex does not form an integral part of this Recommendation | International Standard)

This annex illustrates the use of the basic and canonical XML Encoding Rules specified in this Recommendation | International Standard by showing XML Markup representations of a (hypothetical) personnel record which is defined using ASN.1.

A.1 ASN.1 description of the record structure

The structure of the hypothetical personnel record is formally described below using ASN.1 specified in ITU-T Rec. X.680 | ISO/IEC 8824-1. This is identical to the example defined in Annex A of ITU-T Rec. X.690 | ISO/IEC 8825-1.

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

NOTE – Tags are used in this example only because it was felt appropriate to use the identical example to that which appeared in the earliest version of ITU-T Rec. X.680 | ISO/IEC 8824-1. They have no effect on BASIC-XER, CXER and EXTENDED-XER encodings.

A.2 ASN.1 description of a record value

The value of John Smith's personnel record is formally described below using the basic ASN.1 value notation:

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

A.3 Basic XML representation of this record value

The representation of the record value given above (after applying the basic XML Encoding Rules defined in this Recommendation | International Standard) is shown below assuming an empty prolog.

The length of this encoding in BASIC-XER is 653 octets ignoring all "white-space". For comparison, the same PersonnelRecord value encoded with the UNALIGNED variant of PER (see ITU-T Rec. X.690 | ISO/IEC 8825-1) is 84 octets, with the ALIGNED variant of PER it is 94 octets, with BER (see ITU-T Rec. X.691 | ISO/IEC 8825-2) using the definite length form it is a minimum of 136 octets, and with BER using the indefinite length form it is a minimum of 161 octets.

```
<PersonnelRecord>
      <name>
            <givenName>John</givenName>
            <initial>P</initial>
            <familyName>Smith</familyName>
      </name>
      <title>Director</title>
      <number>51</number>
      <dateOfHire>19710917</dateOfHire>
      <nameOfSpouse>
            <givenName>Mary</givenName>
            <initial>T</initial>
            <familyName>Smith</familyName>
      </nameOfSpouse>
      <children>
            <ChildInformation>
                  <name>
                        <givenName>Ralph</givenName>
                        <initial>T</initial>
                        <familyName>Smith</familyName>
                  </name>
                  <dateOfBirth>19571111</dateOfBirth>
            </ChildInformation>
            <ChildInformation>
                  <name>
                        <givenName>Susan</givenName>
                        <initial>B</initial>
                        <familyName>Jones</familyName>
                  </name>
                  <dateOfBirth>19590717</dateOfBirth>
            </ChildInformation>
      </children>
</PersonnelRecord>
```

A.4 Canonical XML representation of this record value

The representation of the record value given above (after applying the Canonical XML Encoding Rules defined in this Recommendation | International Standard) is shown below:

```
<PersonnelRecord><name><givenName>John</givenName><initial>P</initial><familyNam
e>Smith</familyName></name><number>51</number><title>Director</title><dateOfHire
>19710917</dateOfHire><nameOfSpouse><givenName>Mary</givenName><initial>T</initi
al><familyName>Smith</familyName></nameOfSpouse><children><ChildInformation><nam
e><givenName>Ralph</givenName><initial>T</initial><familyName>Smith</familyName>
</name><dateOfBirth>19571111</dateOfBirth></ChildInformation><ChildInformation><
name><givenName>Susan</givenName><initial>B</initial><familyName>Jones</familyNa
me></name><dateOfBirth>19590717</dateOfBirth></ChildInformation></children></Per
sonnelRecord>
```

Insert a new Annex B as follows:

Annex B

Partial XML content and deterministic encodings

(This annex does not form an integral part of this Recommendation | International Standard)

B.1 Partial XML content

NOTE – This annex describes validity when **MODIFIED-ENCODINGS** is in use.

B.1.1 The following subclauses describe the construction of partial XML element content. This clause describes what partial XML element content is produced as part of encodings, and B.2 specifies restrictions on partial XML element content that are necessary to satisfy the requirement of 10.2.11. If an ASN.1 specification with XER encoding instructions does not violate these restrictions, it is a legal specification, and tools can easily check its legality. If the restrictions **are** violated, the specification may still not violate the normative requirements of 10.2.11, but tools may find it hard to check that this is the case.

NOTE – The restrictions are designed to ensure that a decoder can easily and unambiguously recover the abstract values that were used by an encoder in the production of the encoding.

B.1.2 A partial XML element content is made up of a combination of single XML elements provided by an **[ELEMENT] SEQUENCE**, **SET**, **SEQUENCE OF**, **SET OF** or **CHOICE**, and of other partial XML element content provided by **[UNTAGGED] SEQUENCE**, **SET**, **SEQUENCE OF**, **SET OF** or **CHOICE**.

NOTE – The boundary between partial XML element content within a larger partial XML element content is not visible in the encoding, but can be determined from the ASN.1 schema and restrictions on the names of elements.

B.1.3 A partial XML element content consists of either:

a) a single XML element; or

b) a concatenation group, consisting of an ordered concatenation of zero, one or more partial XML element content in which some of the partial XML element content may be absent in an instance of encoding (representing the absence of an optional abstract value); or

NOTE – An encoding of an **[UNTAGGED] SEQUENCE** or **SET** type will in general produce a concatenation group.

c) a repetition group, consisting of the repetition (unlimited or constrained) of partial XML element content (called the repeated component) produced from the component of a **SEQUENCE OF** or **SET OF**; or

NOTE – An encoding of an **[UNTAGGED] SEQUENCE OF** or **SET OF** type will in general produce a repetition group.

d) an alternatives group, consisting of the presence of a single partial XML element content chosen from a set of alternative partial XML element contents (of which exactly one is present in an encoding).

NOTE – An encoding of a **CHOICE** type produces an alternatives group. Each alternative of the **CHOICE** type produces one of the alternative partial XML element contents for that **CHOICE** type.

B.2 Recommended restrictions on encodings producing partial XML element content

B.2.1 For the purposes of this clause only, any repetition group is treated as if it were optional, that is, may have zero repetitions.

NOTE – The restriction that the repetition group be treated as if it were optional is not strictly necessary if there are constraints that require at least one repetition of the corresponding ASN.1 type, but is introduced for simplicity.

B.2.2 For the purposes of this clause only, a requirement that element names be distinct should be interpreted as follows:

a) all comparisons are made after the application of any final **NAME** and **NAMESPACE** encoding instructions on the type that generated the name;

b) names that are namespace-qualified names are distinct from unqualified names;

c) namespace-qualified names are distinct if and only if they differ in either their unprefixed name or their namespace name or both.

B.2.3 For any resulting partial XML element content, there is possible ambiguity (and hence a possible violation of 10.2.11) if the conditions specified in this subclause B.2 are not satisfied for all possible choices of alternatives in an alternatives group, for all possible exercise of optionality in a concatenation group, for all possible repetitions of a repeated group, and for all possible ordering of the encodings of the components of a set.

NOTE – In reading and implementing the following clauses, the above text saying "for all possible" is very important. Implementers of tools that determine what is an unambiguous specification and what is not will need to analyse all possible combinations of choices, optionality, repetitions and ordering.

B.2.4 (Delimitation requirement) There should be no two adjacent partial XML element contents with the same element name for the first element of the second partial XML element content and for the last element of the first partial XML element content, unless the first partial XML element content is self-delimiting.

EXAMPLE 1: Partial XML element content produced by an **[UNTAGGED] SEQUENCE** is self-delimiting if it does not end with an **OPTIONAL** element.

EXAMPLE 2: Partial XML element content produced by an **[UNTAGGED] SEQUENCE OF** is self-delimiting if it has a fixed number of iterations, which themselves are self-delimiting. This means, inter alia, that **SEQUENCE OF** [**UNTAGGED**] **SEQUENCE OF INTEGER** is ambiguous and violates 10.2.11 unless the number of repetitions of the second **SEQUENCE OF** is fixed.

EXAMPLE 3: Partial XML element content produced by an [**UNTAGGED**] **SET** is never self-delimiting if it has any optional elements.

B.2.5 (Alternative determination requirement) The first XML elements of the alternative partial XML element content in an alternatives group should all have distinct element names.

NOTE – The above text ignores the possible use of **USE-TYPE** and **USE-UNION**, which are beyond the scope of this annex.

EXAMPLE 4: An encoding of:

```
BadExample1 ::= CHOICE {
      -- First alternative partial XML element content
      alt1  [UNTAGGED] SEQUENCE {
            name       UTF8String,
            zip-code   UTF8String },
      alt2 [UNTAGGED] SEQUENCE {
            name       UTF8String,
            post-code  UTF8String } }
```

is not in fact an ambiguous EXTENDED-XER encoding (for a human decoder), but it violates the above requirement and also violates 10.2.11. It is an illegal use of encoding instructions.

B.2.6 (Optionality determination requirement) The XML element names of the first XML element of all consecutive optional partial XML element content plus that of the next following mandatory partial XML element content should be distinct.

NOTE – This means, inter alia, that any optional partial XML element content at the end of a group that is being repeated and any optional partial XML element content at its start have to have distinct XML element names unless the number of repetitions is restricted to a maximum of 1. If the entire partial XML element content of the group that is being repeated is optional, then their XML element names should all be distinct.

EXAMPLE 5: An encoding of:

```
BadExample2 ::= SEQUENCE OF {
            [UNTAGGED] SEQUENCE {
                  first       [UNTAGGED] CommonInitialParms,
                  second      MainInformation,
                  third       [UNTAGGED] CommonEndParms } }
```

where

```
CommonInitialParms ::= SEQUENCE {  date  GeneralizedTime OPTIONAL,
                                   married BOOLEAN}
CommonEndParms ::=      SEQUENCE { name  UTF8String,
                                   date GeneralizedTime OPTIONAL}
```

violates the optionality determination requirement and also violates 10.2.11. It is an illegal use of encoding instructions.

B.2.7 (Repetition count determination requirement) All repetition groups that have a number of repetitions that is not fixed should be followed by a partial XML element content whose first XML element has a name that is distinct from the name of the first XML element of the partial XML element content that is being repeated.

EXAMPLE 6: An encoding of:

```
BadExample3 ::=   SEQUENCE {
      required-items   [UNTAGGED] SEQUENCE OF Book,
      optional-items   [UNTAGGED] SEQUENCE OF Book }
```

violates the repetition count determination requirement and also violates 10.2.11. It is an illegal use of encoding instructions. Alternatively:

```
GoodExample1     ::=   SEQUENCE {
     required-items   [UNTAGGED] SEQUENCE OF required-books Book ,
     optional-items   [UNTAGGED] SEQUENCE OF optional-books Book }
```

would be a legal use of encoding instructions.

B.2.8 (Set component determination requirement) The first XML element in the partial XML content of the components of a concatenation group that is an encoding of a set type should have an XML element name that is distinct from the name of the first XML element in the partial XML content of all other components.

EXAMPLE 7: An encoding of:

```
BadExample4 ::=   SET {
     uk-mailing  UNTAGGED] SEQUENCE {name UTF8String, post-code UTF8String}
     us-mailing  [UNTAGGED] SEQUENCE {name UTF8String, zip-code UTF8String}}
```

violates the component determination requirement and also violates 10.2.11. It is an illegal use of encoding instructions. Alternatively:

```
GoodExample2     ::=   SET {
     uk-mailing  UNTAGGED] SEQUENCE {uk-name UTF8String, post-code UTF8String}
     us-mailing  [UNTAGGED] SEQUENCE {us-name UTF8String, zip-code UTF8String}}
```

would be a legal use of encoding instructions.

Insert a new Annex C as follows:

Annex C

Examples of EXTENDED-XER encodings using XER encoding instructions

(This annex does not form an integral part of this Recommendation | International Standard)

C.1 Introduction

C.1.1 This annex provides tutorial information and examples on the application of XER encoding instructions.

NOTE – All ASN.1 examples in this annex assume an environment of **AUTOMATIC TAGS**.

C.1.2 Encoding instructions normally need to be assigned to an ASN.1 specification only if the designer has a requirement for the actual form of the XML encoding to match that defined by other schema specifications, or expected by other XML tools. Otherwise, ASN.1 alone (with BASIC-XER or CXER encoding) can be used.

C.1.3 If ASN.1 is used as the schema definition notation, then additional use of encoding instructions will in general provide more compact XML encodings than use of ASN.1 alone, but the encodings are still far more verbose than use of ASN.1 with PER.

NOTE – The examples (and the identifiers and type names used) are designed to illustrate features of EXTENDED-XER, and do not in general represent real-world specifications.

C.1.4 XER encoding instructions broadly fall into two categories.

C.1.5 The first category is encoding instructions that are likely to be generally useful in designing the form of an XML document. These are generally allowed even when **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** is absent. The two most useful of these are **ATTRIBUTE** and **LIST**, and C.2 provides simple examples of their use.

C.1.6 The second category is encoding instructions that are designed to support the mapping from W3C XML Schema specified in ITU-T Rec. X.694 | ISO/IEC 8825-5. These generally require the presence of **GLOBAL-DEFAULTS MODIFIED-ENCODINGS** in an Encoding Control Section, but that is not shown in the examples. In these examples, any type reference commencing with "XSD." is assumed to be imported from Annex A of ITU-T Rec. X.694 | ISO/IEC 8825-5. C.3 provides examples of their use. **These examples are not complete ASN.1 modules, nor are they complete XML documents**: module headers are generally omitted; and any XML attribute commencing "**asn1:**" is assumed to be a control attribute using the asn1 namespace for the control attribute, where the prefix "**asn1**" is assumed to be already declared. (In practice, if the encoding is derived from W3C XML Schema, the prefix "**xsi**" is more likely to be used, with the XSI namespace.)

C.1.7 In almost all cases, prefixed encoding instructions are used for clarity, although in a real specification greater brevity (and a clearer separation of abstract syntax definition from encoding issues) will be obtained by the use of an Encoding Control Section.

C.2 Simple examples

C.2.1 A base-ball card

```
BBCard ::= SEQUENCE {
      name   [ATTRIBUTE] IA5String,
      team   [ATTRIBUTE] IA5String,
      age    INTEGER,
      position    IA5String,
      handedness ENUMERATED {
      left-handed,
      right-handed,
      ambidextrous },
      batting-average REAL }
```

Ignoring the encoding instructions (BASIC-XER), we could get:

```
<BBCard>
      <name>Jorge Posada</name>
      <team>New York Yankees</team>
      <age>29</age>
      <position>C</position>
      <handedness><right-handed/></handedness>
      <batting-average>0.277</batting-average>
</BBCard>
```

The EXTENDED-XER encoding (with MODIFIED-ENCODINGS) of the same value is:

```
<BBCard name = “Jorge Posada” team = “New York Yankees” >
      <age>29</age>
      <position>C</position>
      <handedness>right-handed</handedness>
      <batting-average>0.277</batting-average>
</BBCard>
```

C.2.2 An employee

```
Employee ::= [NAME AS UNCAPITALIZED] SEQUENCE {
      id          [ATTRIBUTE] INTEGER(0..MAX),
      recruited   XSD.Date,
      salaries    [LIST] SEQUENCE
                      OF salary REAL }
```

Ignoring the encoding instructions (BASIC-XER), we could get:

```
<Employee>
      <id>239</id>
      <recruited>27-11-2002</recruited>
      <salaries>
           <salary>29876</salary>
           <salary>54375</salary>
           <salary>98435</salary>
      </salaries>
</Employee>
```

The EXTENDED-XER encoding of the same value is:

```
<employee id = "239">
      <recruited>27-11-2002</recruited>
      <salaries>29876 54375 98435</salaries>
</employee>
```

Using an XER Encoding Control Section, we would have:

```
Employee ::= SEQUENCE {
      id          INTEGER(0..MAX),
      recruited   Date,
      salaries    SEQUENCE
             OF salary REAL }

ENCODING-CONTROL XER
       NAME Employee AS UNCAPITALIZED
           ATTRIBUTE Employee.id
           LIST Employee.salaries
```

C.3 More complex examples

C.3.1 Using a union of two simple types

```
Int-or-boolean ::= [USE-UNION] CHOICE {
      int         INTEGER,
      boolean     BOOLEAN }
```

Encodings could be:

```
<Int-or-boolean><int>39</int></Int-or-boolean>                -- BASIC-XER
<Int-or-boolean><boolean><true/></boolean></Int-or-boolean>   -- BASIC-XER
<Int-or-boolean>39</Int-or-boolean>                           -- EXTENDED-XER
<Int-or-boolean>true</Int-or-boolean>                         -- EXTENDED-XER
```

C.3.2 Using a type identification attribute

```
Int-or-boolean ::= [USE-TYPE] CHOICE {
      int         INTEGER,
      boolean     BOOLEAN }
```

Encodings could be:

```
<Int-or-boolean><int>39</int></Int-or-boolean>              -- BASIC-XER
<Int-or-boolean><boolean><true/></boolean></Int-or-boolean> -- BASIC-XER
<Int-or-boolean asn1:type="int">39</Int-or-boolean>         -- EXTENDED-XER
<Int-or-boolean asn1:type="boolean">true</Int-or-boolean>   -- EXTENDED-XER
```

C.3.3 Using enumeration values

```
PrimesUnder30 ::= [USE-NUMBER] ENUMERATED {
                  int2(2), int3(3), int5(5), int7(7), int11(11), int13(13),
                  int17(17), int19(19), int23(23), int29(29)}
InputValues ::= [ATTRIBUTE] [LIST] SEQUENCE OF PrimesUnder30
PrimeProducts ::= SEQUENCE {
            input  InputValues,
            output [ATTRIBUTE] [DECIMAL] REAL}
```

Encodings could be:

```
<PrimeProducts>
     <input><int2/><int7/><int17/><int23/><int29/><int3/></input>
     <output>476338.00</output>
</PrimeProducts>                                  -- BASIC-XER
<PrimeProducts input="2  7  17  23  29  3" output="476338.00"/>
                                                  -- EXTENDED-XER
```

C.3.4 Using an empty encoding for a default value

```
Responses ::= ENUMERATED {ringing, engaged, number-not-known }
CallDetails ::= [DEFAULT-FOR-EMPTY number-not-known] SEQUENCE {
     number     [ATTRIBUTE] NumericString,
     response   Response }
```

Encodings could be:

```
<CallDetails>
     <number>0164593746</number>
     <response><number-not-known/></response>
</CallDetails>                                    -- BASIC-XER
<CallDetails number="0164593746"/>       -- EXTENDED-XER
```

C.3.5 Using embedded-values for notification of a payment due

```
Notification ::= SEQUENCE {
     text [EMBED-VALUES] SEQUENCE OF UTF8String,
     account     INTEGER,
     amount-due  INTEGER,
     payable-by  XSD:Date } (CONSTRAINED BY {/* Shall conform to ITU-T Rec. X.693 |
     ISO/IEC 8825-4, 25.2 */})
```

A value in basic ASN.1 value notation could be:

```
firstNotification Notification ::= {
                  text {"Please note the following details:",
                        "(your business account)",
                         "This is in excess of your normal monthly allowance",
                         "or earlier"},
                  account 568903,
                  amount-due  536,
                  payable-by  "27-08-2003" }
```

The EXTENDED-XER encoding would be:

```
<Notification>
     Please note the following details:
     <account>568903</account>
     (your business account)
     <amount-due>536</amount-due>
     This is in excess of your normal monthly allowance
     <payable-by>27-08-2003</payable-by>
     or earlier
</Notification>
```

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	**Data networks and open system communications**
Series Y	Global information infrastructure, Internet protocol aspects and Next Generation Networks
Series Z	Languages and general software aspects for telecommunication systems

24996

Printed in Switzerland

Geneva, 2004