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

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

Reference number
ISO/IEC 8824-1:1998(E)



PDF disclaimer

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

Adobe is a trademark of Adobe Systems Incorporated.

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

© ISO/IEC 1998

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

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Published by ISO in 1999

Printed in Switzerland

CONTENTS

	<i>Page</i>
1 Scope	1
2 Normative references.....	1
2.1 Identical Recommendations International Standards.....	1
2.2 Paired Recommendations International Standards equivalent in technical content.....	2
2.3 Additional references	2
3 Definitions	3
3.1 Information object specification.....	3
3.2 Constraint specification	3
3.3 Parameterization of ASN.1 specification	3
3.4 Presentation service definition	3
3.5 Presentation protocol specification.....	3
3.6 Structure for identification of organizations.....	3
3.7 Universal Multiple-Octet Coded Character Set (UCS)	3
3.8 Additional definitions.....	4
4 Abbreviations	8
5 Notation	8
5.1 Productions.....	8
5.2 The alternative collections.....	9
5.3 Example of a production	9
5.4 Layout.....	9
5.5 Recursion.....	9
5.6 References to a collection of sequences	9
5.7 References to an item	9
5.8 Short-hand notations	10
6 The ASN.1 model of type extension.....	10
7 Extensibility requirements on encoding rules.....	11
8 Tags	11
9 Use of the ASN.1 notation.....	12
10 The ASN.1 character set.....	13
11 ASN.1 items	13
11.1 General rules	13
11.2 Type references	13
11.3 Identifiers	14
11.4 Value references	14
11.5 Module reference.....	14
11.6 Comment	14
11.7 Empty item	14
11.8 Number item.....	14
11.9 Binary string item.....	14
11.10 Hexadecimal string item.....	15
11.11 Character string item	15
11.12 Assignment item.....	16
11.13 Range separator	16

11.14	Ellipsis	16
11.15	Left version brackets	16
11.16	Right version brackets	16
11.17	Single character items	16
11.18	Reserved words	17
12	Module definition	17
13	Referencing type and value definitions	21
14	Notation to support references to ASN.1 components	22
15	Assigning types and values.....	23
16	Definition of types and values	24
17	Notation for the boolean type	26
18	Notation for the integer type.....	26
19	Notation for the enumerated type	27
20	Notation for the real type.....	28
21	Notation for the bitstring type	29
22	Notation for the octetstring type	30
23	Notation for the null type.....	31
24	Notation for sequence types	31
25	Notation for sequence-of types.....	34
26	Notation for set types.....	34
27	Notation for set-of types	35
28	Notation for choice types.....	35
29	Notation for selection types.....	36
30	Notation for tagged types	37
31	Notation for the object identifier type	38
32	Notation for the embedded-pdv type	39
33	Notation for the external type	41
34	The character string types.....	42
35	Notation for character string types	43
36	Definition of restricted character string types.....	43
37	Naming characters and collections defined in ISO/IEC 10646	47
38	Canonical order of characters	49
39	Definition of unrestricted character string types.....	50
40	Notation for types defined in clauses 41 to 43	52
41	Generalized time.....	52
42	Universal time.....	53
43	The object descriptor type	53
44	Constrained Types	54
45	The exception identifier.....	55
46	Element set specification	55
47	The extension marker	56
48	Subtype elements	58
48.1	General	58
48.2	Single Value	59
48.3	Contained Subtype	59
48.4	Value Range	60

48.5	Size Constraint	60
48.6	Type Constraint	61
48.7	Permitted Alphabet	61
48.8	Inner Subtyping	61
Annex A	– Use of ASN.1-88/90 notation	63
A.1	Maintenance	63
A.2	Mixing ASN.1-88/90 and current ASN.1 notation	63
A.3	Migration to the current ASN.1 notation	63
Annex B	– Assignment of object identifier values	65
Annex C	– Examples and hints	66
C.1	Example of a personnel record	66
C.2	Guidelines for use of the notation	67
C.3	Identifying abstract syntaxes	78
C.4	Subtypes	79
Annex D	– Tutorial annex on ASN.1 character strings	82
D.1	Character string support in ASN.1	82
D.2	The UniversalString, UTF8String and BMPString types	82
D.3	On ISO/IEC 10646-1 conformance requirements	83
D.4	Recommendations for ASN.1 users on ISO/IEC 10646-1 conformance	83
D.5	Adopted subsets as parameters of the abstract syntax	84
D.6	The CHARACTER STRING type	84
Annex E	– Superseded features	86
E.1	Use of identifiers now mandatory	86
E.2	The choice value	86
E.3	The any type	86
E.4	The macro capability	87
Annex F	– Tutorial annex on the ASN.1 model of type extension	88
F.1	Overview	88
F.2	Effects on version numbering, etc.	89
F.3	Requirements on encoding rules	90
Annex G	– Summary of the ASN.1 notation	91

Foreword

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

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 8824 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 8824-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*, in collaboration with ITU-T. The identical text is published as ITU-T Recommendation X.680.

This second edition cancels and replaces the first edition (ISO/IEC 8824-1:1995), which has been technically revised. It also incorporates Amendment 1:1996 and Technical Corrigendum 1:1996.

ISO/IEC 8824 consists of the following parts, under the general title *Information technology — Abstract Syntax Notation One (ASN.1)*:

- *Part 1: Specification of basic notation*
- *Part 2: Information object specification*
- *Part 3: Constraint specification*
- *Part 4: Parameterization of ASN.1 specifications*

Annexes A and B form a normative part of this part of ISO/IEC 8824. Annexes C to G are for information only.

Introduction

This Recommendation | International Standard presents a standard notation for the definition of data types and values. A *data type* (or *type* for short) is a class of information (for example, numeric, textual, still image or video information). A *data value* (or *value* for short) is an instance of such a class. This Recommendation | International Standard defines several basic types and their corresponding values, and rules for combining them into more complex types and values.

Although this standard notation is defined within the OSI framework, it can be used for many other purposes. In the lower layers of the OSI Basic Reference Model (see ITU-T Rec. X.200 | ISO/IEC 7498-1) and in many other protocol architectures, each message is specified as the binary value of a sequence of octets. In the Presentation layer of OSI (see ITU-T Rec. X.216 | ISO/IEC 8822), the nature of user data parameters changes. However, Application layer standards need to define quite complex data types to carry their messages, without concern for their binary representation. In order to specify the data types, they require a notation which does not necessarily determine the representation of each value. Such notation has to be supplemented by the specification of one or more algorithms called **encoding rules** which determine the value of the lower layer octets that carry the Application data (called the **transfer syntax**). The Presentation layer protocol of OSI (see ITU-T Rec. X.226 | ISO/IEC 8823-1) can negotiate which transfer syntaxes (**encodings**) are to be used.

Outside the context of OSI there is increasing recognition of the notion of an abstract value of some class (e.g. a particular 256-colour picture) divorced from the details of any particular encoding where in order to correctly interpret the bit-pattern representation of the value, it is necessary to know (usually from the context), the type (class) of the value being represented, as well as the encoding mechanism being employed. Thus, the identification of a type is an important part of this Recommendation | International Standard.

A very general technique for defining a complicated type at the abstract level is to define a small number of **simple types** by defining all possible values of the simple types, then combining these simple types in various ways. Some of the ways of defining new types are as follows:

- a) given an (ordered) list of existing types, a value can be formed as an (ordered) sequence of values, one from each of the existing types; the collection of all possible values obtained in this way is a new type; (if the existing types in the list are all distinct, this mechanism can be extended to allow omission of some values from the list);
- b) given an unordered set of (distinct) existing types, a value can be formed as an (unordered) set of values, one from each of the existing types; the collection of all possible unordered sets of values obtained in this way is a new type; (the mechanism can again be extended to allow omission of some values);
- c) given a single existing type, a value can be formed as an (ordered) list or (unordered) set of zero, one or more values of the existing type; the collection of all possible lists or sets of values obtained in this way is a new type;
- d) given a list of (distinct) types, a value can be chosen from any one of them; the set of all possible values obtained in this way is a new type;
- e) given a type, a new type can be formed as a subset of it by using some structure or order relationship among the values.

An important aspect of combining types in this way is that encoding rules should recognize the combining constructs, providing unambiguous encodings of the collection of values of the basic types. Thus, every basic type defined using the notation specified in this Recommendation | International Standard is assigned a **tag** to aid in the unambiguous encoding of values.

Four classes of tag are specified in the notation.

The first is the **universal** class. Universal class tags are only used as specified within this Recommendation | International Standard, and each tag is either:

- a) assigned to a single type; or
- b) assigned to a construction mechanism.

Users of this notation are not allowed to explicitly specify universal class tags in their ASN.1 specifications, for these tags are built-in and can be specified explicitly only in this Recommendation | International Standard.

The other three classes of tag are called **application** class tags, **private** class tags, and **context-specific** class tags. There is no formal difference between use of tags from these three classes. Where application class tags are employed, a private or context-specific class tag could generally be applied instead, as a matter of user choice and style. The presence of the three classes is largely for historical reasons, but guidance is given in C.2.12 on the way in which the classes are usually employed.

Tags are mainly intended for machine use, and are not essential for the human notation defined in this Recommendation | International Standard. Where, however, it is necessary to require that certain types be distinct, this is expressed by requiring that they have distinct tags. The allocation of tags is therefore an important part of the use of this notation.

NOTE – Within this Recommendation | International Standard, tag values are assigned to all simple types and construction mechanisms. The restrictions placed on the use of the notation ensure that tags can be used in transfer for unambiguous identification of values.

An ASN.1 specification will initially be produced with a set of fully defined ASN.1 types. At a later stage, however, it may be necessary to change those types (usually by the addition of extra components in a sequence or set type). If this is to be possible in such a way that implementations using the old type definitions can interwork with implementations using the new type definitions in a defined way, encoding rules need to provide appropriate support. The ASN.1 notation supports the inclusion of an **extension marker** on a number of types. This signals to encoding rules the intention of the designer that this type is one of a series of related types (i.e. versions of the same initial type) called an **extension series**, and that the encoding rules are required to enable information transfer between implementations using different types that are related by being part of the same extension series.

Clauses 10 to 31 (inclusive) define the simple types supported by ASN.1, and specify the notation to be used for referencing simple types and for defining new types using them. Clauses 10 to 31 also specify the notation to be used for specifying values of types defined using ASN.1.

Clauses 32 to 33 (inclusive) define the types supported by ASN.1 for carrying within them the complete encoding of ASN.1 types.

Clauses 34 to 39 (inclusive) define the character string types.

Clauses 40 to 43 (inclusive) define certain types which are considered to be of general utility, but which require no additional encoding rules.

Clauses 44 and 48 define a notation which enables subtypes to be defined from the values of a parent type.

Annex A forms an integral part of this Recommendation | International Standard, and gives guidance on how users of this Recommendation | International Standard can refer to ASN.1 types and values defined using CCITT Rec. X.208 | ISO/IEC 8824.

Annex B forms an integral part of this Recommendation | International Standard, and records object identifier and object descriptor values assigned in this Recommendation | International Standard.

Annex C does not form an integral part of this Recommendation | International Standard, and provides examples and hints on the use of the ASN.1 notation.

Annex D does not form an integral part of this Recommendation | International Standard, and provides a tutorial on ASN.1 character strings.

Annex E does not form an integral part of this Recommendation | International Standard, and describes features of the previous version of ASN.1 that have been superseded.

Annex F does not form an integral part of this Recommendation | International Standard, and provides a tutorial on the ASN.1 model of type extension.

Annex G does not form an integral part of this Recommendation | International Standard, and provides a summary of ASN.1 using the notation of clause 5.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

**INFORMATION TECHNOLOGY –
ABSTRACT SYNTAX NOTATION ONE (ASN.1):
SPECIFICATION OF BASIC NOTATION**

1 Scope

This Recommendation | International Standard provides a standard notation called Abstract Syntax Notation One (ASN.1) that is used for the definition of data types, values, and constraints on data types.

This Recommendation | International Standard

- defines a number of simple types, with their tags, and specifies a notation for referencing these types and for specifying values of these types;
- defines mechanisms for constructing new types from more basic types, and specifies a notation for defining such types and assigning them tags, and for specifying values of these types;
- defines character sets (by reference to other Recommendations and/or International Standards) for use within ASN.1;
- defines a number of useful types (using ASN.1), which can be referenced by users of ASN.1;

The ASN.1 notation can be applied whenever it is necessary to define the abstract syntax of information. It is particularly, but not exclusively, applicable to application protocols.

The ASN.1 notation is referenced by other standards which define encoding rules for the ASN.1 types.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.1 Identical Recommendations | International Standards

- ITU-T Recommendation X.200 (1994) | ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*.
- ITU-T Recommendation X.216 (1994) | ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition*.
- ITU-T Recommendation X.226 (1994) | ISO/IEC 8823-1:1994, *Information technology – Open Systems Interconnection – Connection-oriented presentation protocol: Protocol specification*.
- ITU-T Recommendation X.660 (1992)/Amd.2(1997) | ISO/IEC 9834-1:1993/Amd.2:1998, *Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: General procedures (plus Amendments 1 and 2)*.

- ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (1997) | ISO/IEC 8824-3:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (1997) | ISO/IEC 8824-4:1998, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*
- ITU-T Recommendation X.690 (1997) | ISO/IEC 8825-1:1998, *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.691 (1997) | ISO/IEC 8825-2:1998, *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER).*

2.2 Paired Recommendations | International Standards equivalent in technical content

- CCITT Recommendation X.208 (1988), *Specification of Abstract Syntax Notation One (ASN.1).*

ISO/IEC 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1).*

2.3 Additional references

- CCITT Recommendation T.61 (1988), *Character repertoire and coded character sets for the international teletex service.*
- CCITT Recommendation T.100 (1988), *International information exchange for interactive videotex.*
- ITU-T Recommendation T.101 (1994), *International interworking for videotex services.*
- ISO *International Register of Coded Character Sets to be used with Escape Sequences.*
- ISO/IEC 646:1991, *Information technology – ISO 7-bit coded character set for information interchange.*
- ISO/IEC 2022:1994, *Information technology – Character code structure and extension techniques.*
- ISO 6523:1984, *Data interchange – Structures for the identification of organizations.*
- ISO/IEC 7350:1991, *Information technology – Registration of repertoires of graphic characters from ISO 10367.*
- ISO 8601:1988, *Data elements and interchange formats – Information interchange – Representation of dates and times.*
- ISO/IEC 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS – Part 1: Architecture and Basic Multilingual Plane.*
- ISO/IEC 10646-1:1993/Amd.2:1996, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane – Amendment 2: UCS Transformation Format 8 (UTF-8).*

3 Definitions

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

3.1 Information object specification

This Recommendation | International Standard uses the following terms defined in ITU-T Rec. X.681 | ISO/IEC 8824-2:

- a) information object;
- b) information object class;
- c) information object set;
- d) instance-of type;
- e) object class field type.

3.2 Constraint specification

This Recommendation | International Standard uses the following terms defined in ITU-T Rec. X.682 | ISO/IEC 8824-3:

- a) component relation constraint;
- b) table constraint.

3.3 Parameterization of ASN.1 specification

This Recommendation | International Standard uses the following terms defined in ITU-T Rec. X.683 | ISO/IEC 8824-4:

- a) parameterized type;
- b) parameterized value.

3.4 Presentation service definition

This Recommendation | International Standard uses the following terms defined in ITU-T Rec. X.216 | ISO/IEC 8822:

- a) (an) abstract syntax;
- b) abstract syntax name;
- c) defined context set;
- d) presentation data value;
- e) (a) transfer syntax;
- f) transfer syntax name.

3.5 Presentation protocol specification

This Recommendation | International Standard uses the following term defined in ITU-T Rec. X.226 | ISO/IEC 8823-1:

- presentation context identifier.

3.6 Structure for identification of organizations

This Recommendation | International Standard uses the following terms defined in ISO 6523:

- a) issuing organization;
- b) organization code;
- c) International Code Designator.

3.7 Universal Multiple-Octet Coded Character Set (UCS)

This Recommendation | International Standard uses the following terms defined in ISO/IEC 10646-1:

- a) Basic Multilingual Plane (BMP);
- b) cell;
- c) combining character;

- d) graphic symbol;
- e) group;
- f) limited subset;
- g) plane;
- h) row;
- i) selected subset.

3.8 Additional definitions

3.8.1 abstract character: The set of information associated with a cell in a table defining a character repertoire.

NOTE – The information will normally include some or all of the following items:

- a) a graphic symbol;
- b) a character name; or
- c) the definition of functions associated with the character when used in particular environments.

3.8.2 abstract value: A value whose definition is based only on the type, independent of how it is represented in any encoding rule.

NOTE – Use of the term "abstract value" is frequently an assertion that what is being said probably varies based upon the encoding rules used.

3.8.3 ASN.1 character set: The set of characters, specified in clause 10, used in the ASN.1 notation.

3.8.4 ASN.1 specification: A collection of one or more ASN.1 modules.

3.8.5 associated type: A type which is used only for defining the value and subtype notation for a type.

NOTE – Associated types are defined in this Recommendation | International Standard when it is necessary to make it clear that there may be a significant difference between how the type is defined in ASN.1 and how it is encoded. Associated types do not appear in user specifications.

3.8.6 bitstring type: A simple type whose distinguished values are an ordered sequence of zero, one or more bits.

NOTE – Where there is a need to carry embedded encodings of an abstract value, the use of the embedded-pdv type will in general provide a more flexible mechanism for announcement or agreement on the nature of the encodings than the bitstring type.

3.8.7 boolean type: A simple type with two distinguished values.

3.8.8 character: A member of a set of elements used for the organization, control or representation of data.

NOTE – For example, this implies that an accent combining character and lower case 'e' are two characters in the ISO 646 French Version, and not the single character é.

3.8.9 character abstract syntax: Any abstract syntax whose values are specified as the set of character strings of zero, one or more characters from some specified collection of characters.

3.8.10 character repertoire: The characters in a character set without any implication on how such characters are encoded.

3.8.11 character string types: Simple types whose values are strings of characters from some defined character set.

3.8.12 character transfer syntax: Any transfer syntax for a character abstract syntax.

NOTE – ASN.1 does not support character transfer syntaxes which do not encode all character strings as an integral multiple of 8 bits.

3.8.13 choice types: Types defined by referencing a list of distinct types; each value of the choice type is derived from the value of one of the component types.

3.8.14 component type: One of the types referenced when defining a CHOICE, SET, SEQUENCE, SET OF, or SEQUENCE OF.

3.8.15 constraint: A notation which can be used in association with a type, to define a subtype of that type.

3.8.16 control characters: Characters appearing in some character repertoires that have been given a name (and perhaps a defined function in relation to certain environments) but which have not been assigned a graphic symbol, and which are not spacing characters.

NOTE – NEWLINE and TAB are examples of control characters that have been assigned a formatting function in a printing environment. DLE is an example of a control character that has been assigned a function in a communication environment.

3.8.17 Coordinated Universal Time (UTC): The time scale maintained by the Bureau International de l'Heure (International Time Bureau) that forms the basis of a coordinated dissemination of standard frequencies and time signals.

NOTE 1 – The source of this definition is Recommendation 460-2 of the Consultative Committee on International Radio (CCIR). CCIR has also defined the acronym for Coordinated Universal Time as UTC.

NOTE 2 – UTC and Greenwich Mean Time are two alternative time standards which for most practical purposes determine the same time.

3.8.18 element: A member of an element class, distinguishable from all other elements of the same class.

3.8.19 element class: A type (whose elements are its values) or information object class (whose elements are all possible objects of that class).

3.8.20 element set: One or more elements of the same element class.

3.8.21 embedded-pdv type: A type whose set of values is the union of the sets of values in all possible abstract syntaxes. This type is a part of an ASN.1 specification that carries a value whose type may be defined externally to that ASN.1 specification. It also carries an identification of the type of the value being carried as well as an identification of the encoding rule used to encode the value.

3.8.22 encoding: The bit-pattern resulting from the application of a set of encoding rules to a value of a specific abstract syntax.

3.8.23 (ASN.1) encoding rules: Rules which specify the representation during transfer of the values of ASN.1 types. Encoding rules also enable the values to be recovered from the representation, given knowledge of the type.

NOTE – For the purpose of specifying encoding rules, the various referenced type (and value) notations, which can provide alternative notations for built-in types (and values), are not relevant.

3.8.24 enumerated types: Simple types whose values are given distinct identifiers as part of the type notation.

3.8.25 extension addition: One of the added notations in an extension series. For set, sequence and choice types, each extension addition is the addition of either a single extension addition group or a single component type. For enumerated types it is the addition of a single further enumeration. For a constraint it is the addition of a subtype element.

NOTE – Extension additions are both textually ordered (following the extension marker) and logically ordered (having increasing enumeration values, and, in the case of CHOICE alternatives, increasing tags.)

3.8.26 extension addition group: One or more components of a set, sequence or choice type grouped within version brackets. An extension addition group is used to clearly identify the components of a set, sequence or choice type that were added in a particular version of an ASN.1 module.

3.8.27 extension addition type: A type contained within an extension addition group or a single component type that is itself an extension addition (in such case it is not contained within an extension addition group).

3.8.28 extensible constraint: A subtype constraint with an extension marker.

3.8.29 extension insertion point: The location within a type definition where extension additions are inserted. This location is the end of the type notation of the immediately preceding type in the extension series if there is a single ellipsis in the type definition, or immediately before the second ellipsis if there is an extension marker pair in the definition of the type.

3.8.30 extension marker: A syntactic flag (an ellipsis) that is included in all types that form part of an extension series.

3.8.31 extension marker pair: A pair of extension markers between which extension additions are inserted.

3.8.32 extension-related: Two types that have the same extension root, where one was created by adding zero or more extension additions to the other.

3.8.33 extension root: An extensible type that is the first type in an extension series. It carries either the extension marker with no additional notation other than comments and white-space between the extension marker and the matching “}” or “)”, or an extension marker pair with no additional notation other than a single comma, comments and white-space between the extension markers.

NOTE – Only an extension root can be the first type in an extension series.

3.8.34 extension series: A series of ASN.1 types which can be ordered in such a way that each successive type in the series is formed by the addition of text at the extension insertion point.

NOTE – Both nested and unnested types can be extended.

3.8.35 extensible type: A type with an extension marker.

3.8.36 external reference: A type reference, value reference, information object, etc., that is defined in some other module than the one in which it is being referenced, and which is being referred to by prefixing the module name to the referenced item.

EXAMPLE – ModuleName.TypeReference

3.8.37 external type: A type which is a part of an ASN.1 specification that carries a value whose type may be defined externally to that ASN.1 specification. It also carries an identification of the type of the value being carried.

3.8.38 false: One of the distinguished values of the boolean type (see "true").

3.8.39 governing; governor: Relative to some object, object set, value set, value or subtype, the information object class or type which controls its interpretation by restricting the items(s) involved to be value notation of that class or type, respectively.

3.8.40 integer type: A simple type with distinguished values which are the positive and negative whole numbers, including zero (as a single value).

NOTE – Particular encoding rules limit the range of an integer, but such limitations are chosen so as not to affect any user of ASN.1.

3.8.41 items: Named sequences of characters from the ASN.1 character set, specified in clause 11, which are used to form the ASN.1 notation.

3.8.42 module: One or more instances of the use of the ASN.1 notation for type, value, etc., encapsulated using the ASN.1 module notation (see clause 12).

3.8.43 null type: A simple type consisting of a single value, also called null.

3.8.44 object: A well-defined piece of information, definition, or specification which requires a name in order to identify its use in an instance of communication.

3.8.45 object descriptor type: A type whose distinguished values are human-readable text providing a brief description of an object.

NOTE – An object descriptor value is usually associated with a single object. Only an object identifier value unambiguously identifies an object.

3.8.46 object identifier: A value (distinguishable from all other such values) which is associated with an object.

3.8.47 object identifier type: A simple type whose distinguished values are the set of all object identifiers allocated in accordance with the rules of ITU-T Rec. X.660 | ISO/IEC 9834-1.

NOTE – The rules of ITU-T Rec. X.660 | ISO/IEC 9834-1 permit a wide range of authorities to independently associate object identifiers with objects.

3.8.48 octetstring type: A simple type whose distinguished values are an ordered sequence of zero, one or more octets, each octet being an ordered sequence of eight bits.

3.8.49 open type notation: An ASN.1 notation used to denote a set of values from more than one ASN.1 type.

NOTE 1 – The term "open type" is used synonymously with "open type notation" in the body of this Recommendation | International Standard.

NOTE 2 – All ASN.1 encoding rules provide unambiguous encodings for the values of a single ASN.1 type. They do not necessarily provide unambiguous encodings for "open type notation", which carries values from ASN.1 types that are not normally determined at specification time. Knowledge of the type of the value being encoded in the "open type notation" is needed before the abstract value for that field can be unambiguously determined.

NOTE 3 – The only notation in this Recommendation | International Standard which is an open type notation is the "ObjectClassFieldType" specified in ITU-T Rec. X.681 | ISO/IEC 8824-2, where the "FieldName" denotes either a type field or a variable-type value field. The "ANY" notation which was defined in CCITT Rec. X.208 | ISO/IEC 8824 was an open type notation.

3.8.50 parent type (of a subtype): The type that is being constrained when defining a subtype.

NOTE – The parent type may itself be a subtype of some other type.

3.8.51 production: A part of the formal notation used to specify ASN.1.

3.8.52 real type: A simple type whose distinguished values (specified in clause 20) are members of the set of real numbers.

3.8.53 recursive definitions: A set of ASN.1 definitions which cannot be reordered so that all types used in a construction are defined before the definition of the construction.

NOTE – Recursive definitions are allowed in ASN.1: the user of the notation has the responsibility for ensuring that those values (of the resulting types) which are used have a finite representation.

3.8.54 restricted character string type: A character string type whose characters are taken from a fixed character repertoire identified in the type specification.

3.8.55 selection types: Types defined by reference to a component type of a choice type, and whose values are precisely the values of that component type.

3.8.56 sequence types: Types defined by referencing an ordered list of types (some of which may be declared to be optional); each value of the sequence type is an ordered list of values, one from each component type.

NOTE – Where a component type is declared to be optional, a value of the sequence type need not contain a value of that component type.

3.8.57 sequence-of types: Types defined by referencing a single component type; each value in the sequence-of type is an ordered list of zero, one or more values of the component type.

3.8.58 set types: Types defined by referencing a fixed, unordered, list of distinct types (some of which may be declared to be optional); each value in the set type is an unordered list of values, one from each of the component types.

NOTE – Where a component type is declared to be optional, the new type need not contain a value of that component type.

3.8.59 set-of types: Types defined by referencing a single component type; each value in the set-of type is an unordered list of zero, one or more values of the component type.

3.8.60 simple types: Types defined by directly specifying the set of its values.

3.8.61 spacing character: A character in a character repertoire which is intended for inclusion with graphic characters in the printing of a character string but which is represented in the physical rendition by empty space; it is not normally considered to be a control character (see 3.8.16).

NOTE – There may be a single spacing character in the character repertoire, or there may be multiple spacing characters with varying widths.

3.8.62 subtype (of a parent type): A type whose values are a subset (or the complete set) of the values of some other type (the parent type).

3.8.63 tag: A type denotation which is associated with every ASN.1 type.

3.8.64 tagged types: A type defined by referencing a single existing type and a tag; the new type is isomorphic to the existing type, but is distinct from it.

3.8.65 tagging: Replacing the existing (possibly the default) tag of a type by a specified tag.

3.8.66 true: One of the distinguished values of the boolean type (see "false").

3.8.67 type: A named set of values.

3.8.68 type reference name: A name associated uniquely with a type within some context.

NOTE – Reference names are assigned to the types defined in this Recommendation | International Standard; these are universally available within ASN.1. Other reference names are defined in other Recommendations | International Standards, and are applicable only in the context of that Recommendation | International Standard.

3.8.69 unrestricted character string type: A type whose values are values from a character abstract syntax identified separately for each instance of use of that type.

3.8.70 user (of ASN.1): The individual or organization that defines the abstract syntax of a particular piece of information using ASN.1.

3.8.71 value: A distinguished member of a set of values.

3.8.72 value reference name: A name associated uniquely with a value within some context.

3.8.73 value set: A collection of values of a type. Semantically equivalent to a subtype.

3.8.74 version brackets: A pair of adjacent left and right brackets ([[or]]) used to delineate the start and end of an extension addition group.

3.8.75 white-space: Any formatting action that yields a space on a printed page, such as the SPACE or TAB character, or multiple uses of such characters.

4 Abbreviations

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

ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules of ASN.1
DCC	Data Country Code
DNIC	Data Network Identification Code
ICD	International Code Designator
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
PDV	Presentation Data Value
PER	Packed Encoding Rules of ASN.1
ROA	Recognized Operating Agency
UCS	Universal Multiple-Octet Coded Character Set

5 Notation

The ASN.1 notation consists of a sequence of characters from the ASN.1 character set specified in clause 10.

Each use of the ASN.1 notation contains characters from the ASN.1 character set grouped into items. Clause 11 specifies all the sequences of characters forming ASN.1 items, and names each item.

The ASN.1 notation is specified in clause 12 (and following clauses) by specifying the collection of sequences of items which form valid instances of the ASN.1 notation, and by specifying the semantics of each sequence.

In order to specify these collections, this Recommendation | International Standard uses a formal notation defined in the following subclauses.

5.1 Productions

A new (more complex) collection of ASN.1 sequences is defined by means of a production. This uses the names of collections of production sequences defined in this Recommendation | International Standard and forms a new collection of production sequences by specifying either:

- that the new collection of production sequences is to consist of any sequence contained in any of the original collections; or
- that the new collection is to consist of any production sequence which can be generated by taking exactly one production sequence from each collection, and juxtaposing them in a specified order.

Each production consists of the following parts, on one or several lines, in order:

- a name for the new collection of production sequences;
- the characters

::=

- one or more alternative collections of production sequences, defined as in 5.2, separated by the character
|

A production sequence is present in the new collection if it is present in one or more of the alternative collections. The new collection is referenced in this Recommendation | International Standard by the name in a) above.

NOTE – If the same production sequence appears in more than one alternative, any semantic ambiguity in the resulting notation is resolved by other parts of the complete ASN.1 production sequence.

5.2 The alternative collections

Each of the alternative collections of production sequences in "one or more alternative collections of" [see 5.1.c)] is specified by a list of names. Each name is either the name of an item, or is the name of a collection of production sequences defined by a production in this Recommendation | International Standard.

The collection of production sequences defined by the alternative consists of all production sequences obtained by taking any one of the production sequences (or the item) associated with the first name, in combination with (and followed by) any one of the production sequences (or item) associated with the second name, in combination with (and followed by) any one of the production sequences (or item) associated with the third name, and so on up to and including the last name (or item) in the alternative.

5.3 Example of a production

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

is a production which associates with the name "BitStringValue" the following production sequences:

- a) any "bstring" (an item); or
- b) any "hstring" (an item); or
- c) any production sequence associated with "IdentifierList", preceded by a "{" and followed by a "}".

NOTE – "{" and "}" are the names of items containing the single characters { and } (see 11.17).

In this example, "IdentifierList" would be defined by a further production, either before or after the production defining "BitStringValue".

5.4 Layout

Each production used in this Recommendation | International Standard is preceded and followed by an empty line. Empty lines do not appear within productions. The production may be on a single line, or may be spread over several lines. Layout is not significant.

5.5 Recursion

The productions in this Recommendation | International Standard are frequently recursive. In this case the productions are to be continuously reapplied until no new sequences are generated.

NOTE – In many cases, such reapplication results in an unbounded collection of allowed sequences, some or all of which may themselves be unbounded. This is not an error.

5.6 References to a collection of sequences

This Recommendation | International Standard references a collection of sequences (part of the ASN.1 notation) by referencing the first name (that appears before the "::=") in a production; the name is surrounded by the double-quote character (") to distinguish it from natural language text, unless it appears as part of a production.

5.7 References to an item

This Recommendation | International Standard references an item by referencing the name of the item; the name is surrounded by the double-quote character (") to distinguish it from natural language text, unless it appears as part of a production and is not a single character item, ":", ".", or "...".

5.8 Short-hand notations

In order to make productions more concise and more readable, the following short-hand notations are used in the definition of the collections of ASN.1 production sequences in ITU-T Rec. X.681 | ISO/IEC 8824-2, ITU-T Rec. X.682 | ISO/IEC 8824-3 and ITU-T Rec. X.683 | ISO/IEC 8824-4 (it is not used anywhere in this Recommendation | International Standard):

- a) An asterisk (*) following two names, "A" and "B", denotes the empty item (see 11.7), or a production sequence associated with "A", or an alternating series of production sequences associated with "A" and "B" both starting and finishing with one associated with "A". Thus:

C ::= A B *

is equivalent to:

C ::= D | empty

D ::= A | A B D

"D" being an auxiliary name not appearing elsewhere in the productions.

EXAMPLE – "C ::= A B *" is the shorthand notation for the following alternatives of C:

empty

A

A B A

A B A B A

A B A B A B A

...

- b) A plus sign(+) is similar to the asterisk in a), except that the empty item is excluded. Thus:

E ::= A B +

is equivalent to:

E ::= A | A B E

EXAMPLE – "E ::= A B +" is the shorthand notation for the following alternatives of E:

A

A B A

A B A B A

A B A B A B A

...

- c) A question mark (?) following a name denotes either the empty item (see 11.7) or a production sequence associated with "A". Thus:

F ::= A ?

is equivalent to:

F ::= empty | A

6 The ASN.1 model of type extension

When decoding an extensible type, a decoder may detect:

- the absence of expected extension additions in a sequence or set type; or
- the presence of arbitrary unexpected extension additions above those defined (if any) in a sequence or set type, or of an unknown alternative in a choice type, or an unknown enumeration in an enumerated type, or of an unexpected length or value of a type whose constraint is extensible.

In formal terms, an abstract syntax defined by the extensible type "X" contains not only the values of type "X", but also the values of all types that are extension-related to "X". Thus, the decoding process never signals an error when either of the above situations (a or b) is detected. The action that is taken in each situation is a matter for the application layer designer to specify.

NOTE – Frequently the action will be to ignore the presence of unexpected additional extensions, and to use a default value or a "missing" indicator for expected extension additions that are absent.

Unexpected extension additions detected by a decoder in an extensible type can later be included in a subsequent encoding of that type (for transmission back to the sender, or to some third party), provided that the same transfer syntax is used on the subsequent transmission.

7 Extensibility requirements on encoding rules

7.1 All ASN.1 encoding rules shall allow the encoding of values of an extensible type "X" in such a way that they can be decoded using an extensible type "Y" that is extension-related to "X". Further, the encoding rules shall allow the values that were decoded using "Y" to be re-encoded (using "Y") and decoded using a third extensible type "Z" that is extension related to "Y" (and hence "X" also).

NOTE – Types "X", "Y" and "Z" may appear in any order in the extension series.

If a value of an extensible type "X" is encoded and then relayed (directly or through a relaying application using extension-related type "Z") to another application that decodes the value using extensible type "Y" that is extension-related to "X", then the decoder using type "Y" obtains an abstract value composed of:

- a) an abstract value of the extension root type;
- b) an abstract value of each extension addition that is present in both "X" and "Y";
- c) delimited encoding for each extension addition (if any) that is in "X" but not in "Y".

The encodings in c) shall be capable of being included in a later encoding of a value of "Y", if so required by the application. That encoding shall be a valid encoding of a value of "X".

Tutorial example: If system A is using an extensible root type (type "X") that is a sequence type or a set type with an extension addition of an optional integer type, while system B is using an extension-related type (type "Y") that has two extension additions where each is an optional integer type, then transmission by B of a value of "Y" which omits the integer value of the first extension addition and includes the second must not be confused by A with the presence of the first (only) extension addition of "X" that it knows about. Moreover, A must be able to re-encode the value of "X" with a value present for the first integer type, followed by the second integer value received from B, if so required by the application protocol.

7.2 All ASN.1 encoding rules shall specify the encoding and decoding of the value of an enumerated type and a choice type in such a way that if a transmitted value is in the set of extension additions held in common by the encoder and the decoder, then it is successfully decoded, otherwise it shall be possible for the decoder to delimit the encoding of it and to identify it as a value of an (unknown) extension addition.

7.3 All ASN.1 encoding rules shall specify the encoding and decoding of types with extensible constraints in such a way that if a transmitted value is in the set of extension additions held in common by the encoder and the decoder, then it is successfully decoded, otherwise it shall be possible for the decoder to delimit the encoding of and to identify it as a value of an (unknown) extension addition.

In all cases, the presence of extension additions shall not affect the ability to recognize later material when a type with an extension marker is nested inside some other type.

NOTE – All variants of the Basic Encoding Rules of ASN.1 and the Packed Encoding Rules of ASN.1 satisfy all these requirements.

8 Tags

8.1 A tag is specified by giving a class and a number within the class. The class is one of:

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

8.2 The number is a non-negative integer, specified in decimal notation.

Restrictions on tags assigned by the user of ASN.1 are specified in clause 30.

Table 1 summarizes the assignment of tags in the universal class which are specified in this Recommendation | International Standard.

Table 1 – Universal class tag assignments

UNIVERSAL 0	Reserved for use by the encoding rules
UNIVERSAL 1	Boolean type
UNIVERSAL 2	Integer type
UNIVERSAL 3	Bitstring type
UNIVERSAL 4	Octetstring type
UNIVERSAL 5	Null type
UNIVERSAL 6	Object identifier type
UNIVERSAL 7	Object descriptor type
UNIVERSAL 8	External type and Instance-of type
UNIVERSAL 9	Real type
UNIVERSAL 10	Enumerated type
UNIVERSAL 11	Embedded-pdv type
UNIVERSAL 12	UTF8String type
UNIVERSAL 13-15	Reserved for future editions of this Recommendation International Standard
UNIVERSAL 16	Sequence and Sequence-of types
UNIVERSAL 17	Set and Set-of types
UNIVERSAL 18-22, 25-30	Character string types
UNIVERSAL 23-24	Time types
UNIVERSAL 31-...	Reserved for addenda to this Recommendation International Standard

8.3 Some encoding rules require a canonical order for tags. To provide uniformity, a canonical order for tags is defined in 8.4.

8.4 The canonical order for tags is defined as follows:

- a) those elements or alternatives with universal class tags shall appear first, followed by those with application class tags, followed by those with context-specific tags, followed by those with private class tags;
- b) within each class of tags, the elements or alternatives shall appear in ascending order of their tag numbers.

9 Use of the ASN.1 notation

9.1 The ASN.1 notation for a type definition shall be "Type" (see 16.1).

9.2 The ASN.1 notation for a value of a type shall be "Value" (see 16.7).

NOTE – It is not in general possible to interpret the value notation without knowledge of the type.

9.3 The ASN.1 notation for assigning a type to a type reference name shall be either "TypeAssignment" (see 15.1), "ValueSetTypeAssignment" (see 15.4), "ParameterizedTypeAssignment" (see ITU-T Rec. X.683 | ISO/IEC 8824-4, 8.2), or "ParameterizedValueSetTypeAssignment" (see ITU-T Rec. X.683 | ISO/IEC 8824-4, 8.2).

9.4 The ASN.1 notation for assigning a value to a value reference name shall be either "ValueAssignment" (see 15.2) or "ParameterizedValueAssignment" (see ITU-T Rec. X.683 | ISO/IEC 8824-4, 8.2).

9.5 The production alternatives of the notation "Assignment" shall only be used within the notation "ModuleDefinition" (except as specified in Note 2 of 12.1).

10 The ASN.1 character set

10.1 An ASN.1 item shall consist of a sequence of the characters listed in Table 2 except as specified in 10.2 and 10.3.

Table 2 – ASN.1 characters

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

NOTE – Where equivalent derivative standards are developed by national standards bodies, additional characters may appear in the following items:

- typereference (see 11.2);
- identifier (see 11.3);
- valuereference (see 11.4);
- modulereference (see 11.5).

When additional characters are introduced to accommodate a language in which the distinction between upper-case and lower-case letters is without meaning, the syntactic distinction achieved by dictating the case of the first character of certain of the above ASN.1 items has to be achieved in some other way. This is to allow valid ASN.1 specifications to be written in various languages.

10.2 Where the notation is used to specify the value of a character string type, all graphic symbols for the defined character set can appear in the ASN.1 notation, surrounded by the double quote characters (") (see 11.11).

10.3 Additional (arbitrary) graphic symbols may appear in the "comment" item (see 11.6).

10.4 There shall be no significance placed on the typographical style, size, colour, intensity, or other display characteristics.

10.5 The upper and lower-case letters shall be regarded as distinct.

11 ASN.1 items

11.1 General rules

11.1.1 The following subclauses specify the characters in ASN.1 items. In each case the name of the item is given, together with the definition of the character sequences which form the item.

11.1.2 Each item specified in the following subclauses (except "bstring", "hstring" and "cstring") shall appear on a single line, and (except for the "comment", "bstring", "hstring" and "cstring" items) shall not contain white-space (see 11.9, 11.10 and 11.11).

11.1.3 The length of a line is not restricted.

11.1.4 The items in the production sequences specified by this Recommendation | International Standard (the ASN.1 notation) may appear on one line or may appear on several lines, and may be separated by white-space, empty lines or comments.

11.1.5 An item shall be separated from a following item by white-space, newline or comment, if the initial character (or characters) of the following item is a permitted character (or characters) for inclusion at the end of the characters in the earlier item.

11.2 Type references

Name of item – typereference

11.2.1 A "typereference" shall consist of an arbitrary number (one or more) of letters, digits, and hyphens. The initial character shall be an upper-case letter. A hyphen shall not be the last character. A hyphen shall not be immediately followed by another hyphen.

NOTE – The rules concerning hyphen are designed to avoid ambiguity with (possibly following) comment.

11.2.2 A "typereference" shall not be one of the reserved character sequences listed in 11.18.

11.3 Identifiers

Name of item – identifier

An "identifier" shall consist of an arbitrary number (one or more) of letters, digits, and hyphens. The initial character shall be a lower-case letter. A hyphen shall not be the last character. A hyphen shall not be immediately followed by another hyphen.

NOTE – The rules concerning hyphen are designed to avoid ambiguity with (possibly following) comment.

11.4 Value references

Name of item – valuereference

A "valuereference" shall consist of the sequence of characters specified for an "identifier" in 11.2. In analysing an instance of use of this notation, a "valuereference" is distinguished from an "identifier" by the context in which it appears.

11.5 Module reference

Name of item – modulereference

A "modulereference" shall consist of the sequence of characters specified for a "typereference" in 11.2. In analysing an instance of use of this notation, a "modulereference" is distinguished from a "typereference" by the context in which it appears.

11.6 Comment

Name of item – comment

11.6.1 A "comment" is not referenced in the definition of the ASN.1 notation. It may, however, appear at any time between other ASN.1 items, and has no syntactic significance.

NOTE – Nonetheless, in the context of a Recommendation | International Standard that uses ASN.1, an ASN.1 comment may contain normative text related to the application semantics, or constraints on the syntax.

11.6.2 A "comment" shall commence with a pair of adjacent hyphens and shall end with the next pair of adjacent hyphens or at the end of the line, whichever occurs first. A comment shall not contain a pair of adjacent hyphens other than the pair which opens it and the pair, if any, which ends it. It may include graphic symbols which are not in the character set specified in 10.1 (see 10.3).

11.7 Empty item

Name of item – empty

The "empty" item contains no characters. It is used in the notation of clause 5 when alternative sets of production sequences are specified, to indicate that absence of all alternatives is possible.

11.8 Number item

Name of item – number

A "number" shall consist of one or more digits. The first digit shall not be zero unless the "number" is a single digit.

NOTE – The "number" item is always mapped to an integer value by interpreting it as decimal notation.

11.9 Binary string item

Name of item – bstring

A "bstring" shall consist of an arbitrary number (possibly zero) of zeros, ones, white-space or newlines, preceded by a single quote (') and followed by the pair of characters:

'B

White-space and newlines that appear within a binary string item have no significance.

EXAMPLE – '01101100'B

11.10 Hexadecimal string item

Name of item – hstring

11.10.1 An "hstring" shall consist of an arbitrary number (possibly zero) of the characters:

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

or white-space or newlines, preceded by a single quote (') and followed by the pair of characters:

'H

White-space and newlines that appear within a hexadecimal string item have no significance.

EXAMPLE – 'AB0196'H

11.10.2 Each character is used to denote the value of a semi-octet using a hexadecimal representation.

11.11 Character string item

Name of item – cstring

11.11.1 A "cstring" shall consist of an arbitrary number (possibly zero) of graphic symbols and spacing characters from the character set referenced by the character string type, preceded and followed by double quotes ("). If the character set includes a double quote, this character (if present in the character string being represented by the "cstring") shall be represented in the "cstring" by a pair of double quotes on the same line with no intervening spacing character. The "cstring" may span more than one line of text, in which case the character string being represented shall not include spacing characters in the position prior to or following the end of line in the "cstring". White-space that appears immediately prior to or following the end of line in the "cstring" have no significance.

NOTE 1 – The "cstring" can only be used to represent character strings for which every character in the string being represented has either been assigned a graphic symbol, or is a spacing character. Where a character string containing control characters needs to be denoted, alternative ASN.1 syntax is available. (See clause 34.)

NOTE 2 – The character string represented by a "cstring" consists of the characters associated with the printed graphic symbols and spacing characters. Spacing characters immediately preceding or following any end of line in the "cstring" are not part of the character string being represented (they are ignored). Where spacing characters are included in the "cstring", or where the graphic symbols in the character repertoire are not unambiguous the character string denoted by "cstring" may be ambiguous.

EXAMPLE 1 – "屎 屍 市 弑"

EXAMPLE 2 – The "cstring":

"ABCDE FGH

IJK""XYZ"

can be used to represent a character string value of type IA5String. The value represented consists of the characters:

ABCDE FGH IJK"XYZ

where the precise number of spaces intended between E and F can be ambiguous if a proportional spacing font (such as is used above) is used in the specification.

11.11.2 When a character is a combining character it shall be denoted in the "cstring" as an individual character. It shall not be overprinted with the characters with which it combines. (This ensures that the order of combining characters in the string value is unambiguously defined.)

EXAMPLE – The accent combining character and lower case 'e' are two characters in the ISO 646 French Version, and thus in a corresponding "cstring" is written as two characters and not as the single character é.

11.11.3 The "cstring" shall not be used to represent character string values which contain control characters. Only graphic and spacing characters are permitted in it.

11.12 Assignment item

Name of item – "::="

This item shall consist of the sequence of characters:

::=

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

11.13 Range separator

Name of item – ".."

This item shall consist of the sequence of characters:

..

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

11.14 Ellipsis

Name of item – "..."

This item shall consist of the sequence of characters:

...

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

11.15 Left version brackets

Name of item – "[["

This item shall consist of the sequence of characters:

[[

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

11.16 Right version brackets

Name of item – "]]"

This item shall consist of the sequence of characters:

]]

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

11.17 Single character items

Names of items –

"{"

"}"

"<"

">"

"."

"("

")"

"["

"]"

"-" (hyphen)

":"

","

"@"

"|"

"!"

"^"

An item with any of the names listed above shall consist of the single character without the quotation marks.

11.18 Reserved words

Names of reserved words –

ABSENT	END	INSTANCE	REAL
ABSTRACT-SYNTAX	ENUMERATED	INTEGER	SEQUENCE
ALL	EXCEPT	INTERSECTION	SET
APPLICATION	EXPLICIT	ISO646String	SIZE
AUTOMATIC	EXPORTS	MAX	STRING
BEGIN	EXTENSIBILITY	MIN	SYNTAX
BIT	EXTERNAL	MINUS-INFINITY	T61String
BMPString	FALSE	NULL	TAGS
BOOLEAN	FROM	NumericString	TeletexString
BY	GeneralizedTime	OBJECT	TRUE
CHARACTER	GeneralString	ObjectDescriptor	TYPE-IDENTIFIER
CHOICE	GraphicString	OCTET	UNION
CLASS	IA5String	OF	UNIQUE
COMPONENT	IDENTIFIER	OPTIONAL	UNIVERSAL
COMPONENTS	IMPLICIT	PDV	UniversalString
CONSTRAINED	IMPLIED	PLUS-INFINITY	UTCTime
DEFAULT	IMPORTS	PRESENT	UTF8String
DEFINITIONS	INCLUDES	PrintableString	VideotexString
EMBEDDED		PRIVATE	VisibleString
			WITH

Items with the above names shall consist of the sequence of characters in the name, and are reserved character sequences.

NOTE 1 – White-space does not occur in these sequences.

NOTE 2 – The keywords CLASS, CONSTRAINED, INSTANCE, SYNTAX and UNIQUE are not used in this Recommendation | International Standard; they are used in ITU-T Rec. X.681 | ISO/IEC 8824-2, ITU-T Rec. X.682 | ISO/IEC 8824-3 and ITU-T Rec. X.683 | ISO/IEC 8824-4.

12 Module definition

12.1 A "ModuleDefinition" is specified by the following productions:

```

ModuleDefinition ::=
    ModuleIdentifier
    DEFINITIONS
    TagDefault
    ExtensionDefault
    "::~="
    BEGIN
    ModuleBody
    END

ModuleIdentifier ::=
    modulereference
    DefinitiveIdentifier
  
```

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

DefinitiveObjIdComponentList ::=
 DefinitiveObjIdComponent |
 DefinitiveObjIdComponent DefinitiveObjIdComponentList

DefinitiveObjIdComponent ::=
 NameForm |
 DefinitiveNumberForm |
 DefinitiveNameAndNumberForm

DefinitiveNumberForm ::= number

DefinitiveNameAndNumberForm ::= identifier "(" DefinitiveNumberForm ")"

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

ExtensionDefault ::=
 EXTENSIBILITY IMPLIED |
 empty

ModuleBody ::=
 Exports Imports AssignmentList |
 empty

Exports ::=
 EXPORTS SymbolsExported ";" |
 empty

SymbolsExported ::=
 SymbolList |
 empty

Imports ::=
 IMPORTS SymbolsImported ";" |
 empty

SymbolsImported ::=
 SymbolsFromModuleList |
 empty

SymbolsFromModuleList ::=
 SymbolsFromModule |
 SymbolsFromModuleList SymbolsFromModule

SymbolsFromModule ::=
 SymbolList FROM GlobalModuleReference

GlobalModuleReference ::=
 modulereference AssignedIdentifier

AssignedIdentifier ::=
 ObjectIdentifierValue |
 DefinedValue |
 empty

SymbolList ::=
 Symbol |
 SymbolList "," Symbol

Symbol ::=

Reference |
ParameterizedReference

Reference ::=

typereference |
valuereference |
objectclassreference |
objectreference |
objectsetreference

AssignmentList ::=

Assignment |
AssignmentList Assignment

Assignment ::=

TypeAssignment |
ValueAssignment |
ValueSetTypeAssignment |
ObjectClassAssignment |
ObjectAssignment |
ObjectSetAssignment |
ParameterizedAssignment

NOTE 1 – The use of a ParameterizedReference in the EXPORTS and IMPORTS lists is specified in ITU-T Rec. X.683 | ISO/IEC 8824-4.

NOTE 2 – For examples (and for the definition in this Recommendation | International Standard of types with universal class tags), the "ModuleBody" can be used outside of a "ModuleDefinition".

NOTE 3 – "TypeAssignment" and "ValueAssignment" productions are specified in clause 15.

NOTE 4 – The grouping of ASN.1 data types into modules does not necessarily determine the formation of presentation data values into named abstract syntaxes for the purpose of presentation context definition.

NOTE 5 – The value of "TagDefault" for the module definition affects only those types defined explicitly in the module. It does not affect the interpretation of imported types.

NOTE 6 – The character semicolon does not appear in the assignment list specification or any of its subordinate productions, and is reserved for use by ASN.1 tool developers.

12.2 The "TagDefault" is taken as "EXPLICIT TAGS" if it is "empty".

NOTE – Clause 30 gives the meaning of "EXPLICIT TAGS", "IMPLICIT TAGS", and "AUTOMATIC TAGS".

12.3 When the "AUTOMATIC TAGS" alternative of "TagDefault" is selected, automatic tagging is said to be selected for the module, otherwise it is said to be not selected. Automatic tagging is a syntactical transformation which is applied (with additional conditions) to the "ComponentTypeLists" and "AlternativeTypeLists" productions occurring within the definition of the module. This transformation is formally specified by 24.7 to 24.9, 26.3 and 28.2 regarding the notations for sequence types, set types and choice types, respectively.

12.4 The "EXTENSIBILITY IMPLIED" option is equivalent to the textual insertion of an extension marker (...) in the definition of each type in the module for which it is permitted. The location of the implied extension marker is the last position in the type where an explicitly specified extension marker is allowed. The absence of "EXTENSIBILITY IMPLIED" means that extensibility is only provided for those types within the module where an extension marker is explicitly present.

NOTE – "EXTENSIBILITY IMPLIED" affects only types. It has no effect on object sets.

12.5 The "modulereference" appearing in the "ModuleIdentifier" production is called the module name.

NOTE – The possibility of defining a single ASN.1 module by the use of several occurrences of "ModuleBody" assigned the same "modulereference" was (arguably) permitted in earlier specifications. It is not permitted by this Recommendation | International Standard.

12.6 Module names shall be used only once (except as specified in 12.9) within the sphere of interest of the definition of the module.

12.7 If the "DefinitiveIdentifier" is not empty, the denoted object identifier value unambiguously and uniquely identifies the module being defined. No defined value may be used in defining the object identifier value.

NOTE – The question of what changes to a module require a new "DefinitiveIdentifier" is not addressed in this Recommendation | International Standard.

12.8 If the "AssignedIdentifier" is not empty, the "ObjectIdentifierValue" and the "DefinedValue" alternatives unambiguously and uniquely identify the module from which items are being imported. When the "DefinedValue" alternative of "AssignedIdentifier" is used, it shall be a value of type object identifier. Each "valuereference" which textually appears within an "AssignedIdentifier" shall satisfy one of the following rules:

- a) It is defined in the "AssignmentList" of the module being defined, and all "valuereferences" which textually appear on the right side of the assignment statement also satisfy this rule (rule "a") or the next rule (rule "b").
- b) It appears as a "Symbol" in a "SymbolsFromModule" whose "AssignedIdentifier" does not textually contain any "valuereferences".

NOTE – It is recommended that an object identifier be assigned so that others can unambiguously refer to the module.

12.9 The "GlobalModuleReference" in a "SymbolsFromModule" shall appear in the "ModuleDefinition" of another module, except that if it includes a non-empty "DefinitiveIdentifier", the "modulereference" may differ in the two cases.

NOTE – A different "modulereference" from that used in the other module should only be used when symbols are to be imported from two modules with the same name (the modules being named in disregard of 12.6). The use of alternative distinct names makes these names available for use in the body of the module (see 12.15).

12.10 When both a "modulereference" and a non-empty "AssignedIdentifier" are used in referencing a module, the latter shall be considered definitive.

12.11 When the referenced module has a non-empty "DefinitiveIdentifier", the "GlobalModuleReference" referencing that module shall not have an empty "AssignedIdentifier".

12.12 When the "SymbolsExported" alternative of "Exports" is selected:

- a) each "Symbol" in "SymbolsExported" shall satisfy one and only one of the following conditions:
 - i) is only defined in the module being constructed; or
 - ii) only appears exactly once in the "SymbolsImported" alternative of "Imports";
- b) every "Symbol" to which reference from outside the module is appropriate shall be included in the "SymbolsExported" and only these "Symbol"s may be referenced from outside the module; and
- c) if there are no such "Symbol"s, then the empty alternative of "SymbolsExported" (not of "Exports") shall be selected.

12.13 When the "empty" alternative of "Exports" is selected, every "Symbol" defined in the module may be referenced from other modules.

NOTE – The "empty" alternative of "Exports" is included for backwards compatibility.

12.14 Identifiers that appear in a "NamedNumberList", "Enumeration" or "NamedBitList" are implicitly exported if the typereference that defines them is exported or appears as a component (or subcomponent) within an exported type.

12.15 When the "SymbolsImported" alternative of "Imports" is selected:

- a) Each "Symbol" in "SymbolsFromModule" shall either be defined in the module body, or be present in the IMPORTS clause, of the module denoted by the "GlobalModuleReference" in "SymbolsFromModule". Importing a "Symbol" present in the IMPORTS clause of the referenced module is only allowed if there is only one occurrence of the "Symbol" in that clause, and the "Symbol" is not defined in the referenced module.

NOTE 1 – This does not prohibit the same symbol name defined in two different modules from being imported into another module. However, if the same "Symbol" name appears more than once in the IMPORTS clause of module A, that "Symbol" name cannot be exported from A for import to another module B.

- b) If the "SymbolsExported" alternative of "Exports" is selected in the definition of the module denoted by the "GlobalModuleReference" in "SymbolsFromModule" the "Symbol" shall appear in its "SymbolsExported".
- c) Only those "Symbol"s that appear amongst the "SymbolList" of a "SymbolsFromModule" may appear as the symbol in any "External<X>Reference" which has the "modulereference" denoted by the "GlobalModuleReference" of that "SymbolsFromModule" (where <X> is "value", "type", "object", "objectclass", or "objectset").
- d) If there are no such "Symbol"s, then the "empty" alternative of "SymbolsImported" shall be selected.

NOTE 2 – An effect of c) and d) is that the statement "IMPORTS;" implies that the module cannot contain an "External<X>Reference".

- e) All the "SymbolsFromModule" in the "SymbolsFromModuleList" shall include occurrences of "GlobalModuleReference" such that:
- i) the "modulereference" in them are all different from each other and from the "modulereference" associated with the referencing module; and
 - ii) the "AssignedIdentifier", when non-empty, denotes object identifier values which are all different from each other and from the object identifier value (if any) associated with the referencing module.

12.16 When the "empty" alternative of "Imports" is selected, the module may still reference "Symbols" defined in other modules by means of an "External<X>Reference".

NOTE – The "empty" alternative of "Imports" is included for backwards compatibility.

12.17 Identifiers that appear in a NamedNumberList, Enumeration or NamedBitList are implicitly imported if the typereference that defines them is imported or appears as a component (or subcomponent) within an exported type.

12.18 A "Symbol" in a "SymbolsFromModule" may appear in "ModuleBody" as a "Reference". The meaning associated with the "Symbol" is that which it has in the module denoted by the corresponding "GlobalModuleReference".

12.19 Where the "Symbol" also appears in an "AssignmentList" (deprecated), or appears in one or more other instances of "SymbolsFromModule", it shall only be used in an "External<X>Reference". Where it does not so appear, it shall be used directly as a "Reference".

12.20 The various alternatives for "Assignment" are defined in the following clauses in this Recommendation | International Standard, except as noted otherwise:

<i>Assignment alternative</i>	<i>Defining subclause</i>
"TypeAssignment"	15.1
"ValueAssignment"	15.2
"ValueSetTypeAssignment"	15.4
"ObjectClassAssignment"	ITU-T Rec. X.681 ISO/IEC 8824-2, 9.1
"ObjectAssignment"	ITU-T Rec. X.681 ISO/IEC 8824-2, 11.1
"ObjectSetAssignment"	ITU-T Rec. X.681 ISO/IEC 8824-2, 12.1
"ParameterizedAssignment"	ITU-T Rec. X.683 ISO/IEC 8824-4, 8.1

The first symbol of every "Assignment" is one of the alternatives of "Reference", denoting the reference name being defined. In no two assignments within an "AssignmentList" shall the reference names be the same.

13 Referencing type and value definitions

13.1 The defined type and value productions:

```

DefinedType ::=
    Externaltypereference |
    typereference |
    ParameterizedType |
    ParameterizedValueSetType

DefinedValue ::=
    Externalvaluereference |
    valuereference |
    ParameterizedValue
  
```

specify the sequences which shall be used to reference type and value definitions. The type identified by a "ParameterizedType" and "ParameterizedValueSetType", and the value identified by a "ParameterizedValue" are specified in ITU-T Rec. X.683 | ISO/IEC 8824-4.

13.2 Except as specified in 12.18, the "typereference", "valuereference", "ParameterizedType", "ParameterizedValueSetType" or "ParameterizedValue" alternatives shall not be used unless the reference is within the "ModuleBody" in which a type or value is assigned (see 15.1 and 15.2) to the typereference or valuereference.

13.3 The "Externaltypereference" and "Externalvaluereference" shall not be used unless the corresponding "typereference" or "valuereference":

- a) has been assigned a type or value respectively (see 15.1 and 15.2); or
- b) are present in the IMPORTS clause,

within the "ModuleBody" used to define the corresponding "modulereference". Referencing an item in the IMPORTS clause of another module shall only be allowed if there is no more than one occurrence of the "Symbol" in that clause.

NOTE – This does not prohibit the same "Symbol" defined in two different modules from being imported into another module. However, if the same "Symbol" appears more than once in the IMPORTS clause of a module A, then that "Symbol" cannot be referenced using module A in an external reference.

13.4 An external reference shall be used in a module only to refer to an item which is defined in a different module, and is specified by the following productions:

Externaltypereference ::=
modulereference
 "."
typereference

Externalvaluereference ::=
modulereference
 "."
valuereference

NOTE – Additional external reference productions (ExternalClassReference, ExternalObjectReference and ExternalObjectSetReference) are specified in ITU-T Rec. X.681 | ISO/IEC 8824-2.

13.5 When the referencing module is defined using the "SymbolsImported" alternative of "Imports", the "modulereference" in the external reference shall appear in the "GlobalModuleReference" of exactly one of the "SymbolsFromModule" in the "SymbolsImported". When the referencing module is defined using the "empty" alternative of "Imports", the "modulereference" in the external reference shall appear in the "ModuleDefinition" of the module (different from the referencing module) where the "Reference" is defined.

14 Notation to support references to ASN.1 components

14.1 There is a requirement for formal reference to components of ASN.1 types, values, etc. for many purposes. One such instance is the need to write text to identify a specific type within some ASN.1 module. This clause defines a notation which can be used to provide such references.

14.2 The notation enables any component of a set or sequence type (which is either mandatorily or optionally present in the type) to be identified.

14.3 Any part of any ASN.1 type definition can be referenced by use of the "AbsoluteReference" syntactic construct:

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

ItemSpec ::=
typereference |
ItemId "." ComponentId

ItemId ::= ItemSpec
ComponentId ::=
identifier | **number** | **"*"**

NOTE – The AbsoluteReference production is not used elsewhere in this Recommendation | International Standard. It is provided for the purposes stated in 14.1.

14.4 The "GlobalModuleReference" identifies an ASN.1 module (see 12.1).

14.5 The "typereference" references any ASN.1 type defined in the module identified by "GlobalModuleReference".

14.6 The "ComponentId" in each "ItemSpec" identifies a component of the type which has been identified by the "ItemId". It shall be the last "ComponentId" if the component it identifies is not a set, sequence, set-of, sequence-of, or choice type.

14.7 The "identifier" form of "ComponentId" can be used if the parent "ItemId" is a set or sequence type, and is required to be one of the "identifier"s of the "NamedType" in the "ComponentTypeLists" of that set or sequence. It can also be used if the "ItemId" identifies a choice type, and is then required to be one of the "identifier"s of a "NamedType" in the "AlternativeTypeLists" of that choice type. It cannot be used in any other circumstance.

14.8 The number form of "ComponentId" can be used only if the "ItemId" is a sequence-of or set-of type. The value of the number identifies the instance of the type in the sequence-of or set-of, with the value "1" identifying the first instance of the type. The value zero identifies a conceptual integer type component (not explicitly present in transfer, and called **the iteration count**) that contains a count of the number of instances of the type in the sequence-of or set-of that are present in the value of the enclosing type.

14.9 The "*" form of "ComponentId" can be used only if the "ItemId" is a sequence-of or set-of. Any semantics associated with the use of the "*" form of "ComponentId" apply to all components of the sequence-of and set-of.

NOTE – In the following example:

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

the components of "T" could be referenced by text outside an ASN.1 module (or in a comment), such as:

```
if (@M.T.b.0 is odd) then:
    (@M.T.b.* shall be an odd integer)
```

which is used to state that if the number of components in "b" is odd, all components of "b" must be odd.

15 Assigning types and values

15.1 A "typereference" shall be assigned a type by the notation specified by the "TypeAssignment" production:

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

The "typereference" shall not be an ASN.1 reserved word (see 11.18).

15.2 A "valuereference" shall be assigned a value by the notation specified by the "ValueAssignment" production:

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

The "Value" being assigned to the "valuereference" shall be a notation for a value of the type defined by "Type" (as specified in 15.3).

15.3 "Value" is a notation for a value of a type if either:

- "Value" is a "BuiltinValue" notation for the type (see 16.8); or
- "Value" is a "DefinedValue" notation for a value of that type.

15.4 A "typereference" can be assigned a value set by the notation specified by the "ValueSetTypeAssignment" production:

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

This notation assigns to "typereference" the type defined as a subtype of the type denoted by "Type" and which contains exactly the values which are specified in or allowed by "ValueSet". The "typereference" shall not be an ASN.1 reserved word (see 11.18), and may be referenced as a type. "ValueSet" is defined in 15.5.

15.5 A value set governed by some type shall be specified by the notation "ValueSet":

```
ValueSet ::= "{" ElementSetSpecs "}"
```

The value set comprises all of the values, of which there shall be at least one, specified by "ElementSetSpecs" (see clause 46).

16 Definition of types and values

16.1 A type shall be specified by the notation "Type":

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

16.2 The built-in types of ASN.1 are specified by the notation "BuiltinType", defined as follows:

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

The various "BuiltinType" notations are defined in the following clauses (in this Recommendation | International Standard unless otherwise stated):

BitStringType	21
BooleanType	17
CharacterStringType	35
ChoiceType	28
EmbeddedPDVType	32
EnumeratedType	19
ExternalType	33
InstanceOfType	ITU-T Rec. X.681 ISO/IEC 8824-2, Annex C
IntegerType	18
NullType	23
ObjectClassFieldType	ITU-T Rec. X.681 ISO/IEC 8824-2, 14.1
ObjectIdentifierType	31
OctetStringType	22

RealType	20
SequenceType	24
SequenceOfType	25
SetType	26
SetOfType	27
TaggedType	30

16.3 The referenced types of ASN.1 are specified by the notation "ReferencedType":

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

The "ReferencedType" notation provides an alternative means of referring to some other type (and ultimately to a built-in type). The various "ReferencedType" notations, and the way in which the type to which they refer is determined, are specified in the following places in this Recommendation | International Standard unless otherwise stated):

DefinedType	13.1
UsefulType	40.1
SelectionType	29
TypeFromObject	ITU-T Rec. X.681 ISO/IEC 8824-2, clause 15
ValueSetFromObjects	ITU-T Rec. X.681 ISO/IEC 8824-2, clause 15

16.4 The "ConstrainedType" is defined in clause 44.

16.5 This Recommendation | International Standard requires the use of the notation "NamedType" in specifying the components of the set type, sequence type and choice types. The notation for "NamedType" is:

NamedType ::= identifier Type

16.6 The "identifier" is used to unambiguously refer to components of a set type, sequence type or choice type in the value notation and in component relation constraints (see ITU-T Rec. X.682 | ISO/IEC 8824-3). It is not part of the type, and has no effect on the type.

16.7 A value of some type shall be specified by the notation "Value":

Value ::= BuiltinValue | ReferencedValue | ObjectClassFieldValue

NOTE – ObjectClassFieldValue is defined in ITU-T Rec. X.681 | ISO/IEC 8824-2, 14.6.

16.8 Values of the built-in types of ASN.1 can be specified by the notation "BuiltinValue", defined as follows:

BuiltinValue ::=
BitStringValue |
BooleanValue |
CharacterStringValue |
ChoiceValue |
EmbeddedPDVValue |
EnumeratedValue |
ExternalValue |
InstanceOfValue |
IntegerValue |
NullValue |
ObjectIdentifierValue |
OctetStringValue |
RealValue |
SequenceValue |
SequenceOfValue |
SetValue |
SetOfValue |
TaggedValue

Each of the various "BuiltinValue" notations is defined in the same clause as the corresponding "BuiltinType" notation, as listed in 16.2 above.