



BS 6962 : 1988
ISO 8824 : 1987
UDC 681.324.02.071.8

© British Standards Institution. No part of this publication may be photocopied or otherwise reproduced without the prior permission in writing of BSI

British Standard for

Open Systems Interconnection: Specification for Abstract Syntax Notation One (ASN.1)

[ISO title: Information processing systems — Open Systems Interconnection —
Specification of Abstract Syntax Notation One (ASN.1)]

Interconnexion de systèmes ouverts: Notation de syntaxe abstraite numéro 1 (ASN.1) — Spécifications

Kommunikation Offener Systeme: Notation Eins für darstellungsunabhängige Syntax

British Standards Institution

AVA **Documents**
Document Sharing Hub

Contents

	Page		Page
National foreword	1	Section two. Character string types	
Committees responsible	Back cover	29 Definition of character string types	22
Specification		Section three. Useful definitions	
0 Introduction	2	30 Generalized time	24
1 Scope and field of application	4	31 Universal time	24
2 References	4	32 The external type	25
3 Definitions	4	33 The object descriptor type	26
4 Abbreviations	6		
5 Notation used in this International Standard	6	Annexes	
6 Use of the ASN.1 notation	8	A The macro notation	28
Section one. Specification of ASN.1 notation		B ISO assignment of OBJECT IDENTIFIER component values	32
7 The ASN.1 character set	9	C CCITT assignment of OBJECT IDENTIFIER component values	33
8 ASN.1 items	9	D Joint assignment of OBJECT IDENTIFIER component values	34
9 Module definition	11	E Examples and hints	35
10 Referencing type and value definitions	12	F Summary of the ASN.1 notation	46
11 Assigning types and values	12		
12 Definition of types and values	13	Tables	
13 Notation for the boolean type	14	1 Universal class tag assignments	8
14 Notation for the integer type	14	2 ASN.1 characters	9
15 Notation for the bitstring type	14	3 Reserved character sequences	10
16 Notation for the octetstring type	15	4 NumericString	22
17 Notation for the null type	16	5 PrintableString	22
18 Notation for sequence types	16	6 List of character string types	23
19 Notation for sequence-of types	17	7 Sequence specified by items	29
20 Notation for set types	17		
21 Notation for set-of types	17		
22 Notation for choice types	18		
23 Notation for selection types	18		
24 Notation for tagged types	19		
25 Notation for the any type	19		
26 Notation for the object identifier type	20		
27 Notation for character string types	21		
28 Notation for types defined in section three	21		

National foreword

This British Standard has been prepared under the direction of the Information Systems Technology Standards Committee. It is identical with ISO 8824 : 1987 'Information processing systems — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)', published by the International Organization for Standardization (ISO). It supersedes DD 103 : 1986, which is withdrawn.

The UK has actively participated through BSI Technical Committee IST/21 in the development of the international standard.

The text of the international standard is technically aligned with the syntax defined in The International Telegraph and Telephone Consultative Committee (CCITT) Recommendation X.409 : 1984. It is expected that future editions of the ISO standard and the CCITT Recommendations will be technically identical.

CCITT Recommendation X.409 : 1984 also contains encoding rules that are technically aligned with the basic encoding rules for ASN.1 contained in the companion BS 6963.

Up-to-date information on existing and projected British Standards publications for Open Systems Interconnection is included in the 'Basic List' published as part of the BSI Information Technology Services (BITS) initiative. For details contact: Marketing Department, BSI, Linford Wood, Milton Keynes, MK14 6LE.

Terminology and conventions. The text of the international standard has been approved as suitable for publication as a British Standard without deviation. Some terminology and certain conventions are not identical with those used in British Standards; attention is drawn especially to the following.

Wherever the words 'International Standard' appear, referring to this standard, they should be read as 'British Standard'.

Cross-references

International standard	Corresponding British Standard
ISO 2014 : 1976	BS 5249 Representation of elements of data in interchanges using data processing systems Part 1 : 1976 Representation of dates and times (Technically equivalent)
ISO 3166 : 1981	BS 5374 : 1981 Specification for codes for the representation of names of countries (Identical)
ISO 7498 : 1984	BS 6568 Reference model of Open Systems Interconnection Part 1 : 1988 Basic reference model (incorporating connectionless mode transmission) (Identical)
ISO 8825 : 1987	BS 6963 : 1988 Open Systems Interconnection: specification of basic encoding rules for Abstract Syntax Notation One (ASN.1) (Identical)

The Technical Committee has reviewed the provisions of ISO 2375, ISO 3307, ISO 4031, ISO 6523, ISO 8822, ISO 8823, and CCITT Recommendation X.409 : 1984, to which reference is made in the text, and has decided that they are acceptable for use in conjunction with this standard. British Standards identical with ISO 8822 and ISO 8823 are in preparation. A copy of CCITT Recommendation X.409 : 1984 is available from BSI Sales Department, Linford Wood, Milton Keynes, MK14 6LE.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

British Standard for

Open Systems Interconnection: Specification for Abstract Syntax Notation One (ASN.1)

0 Introduction

In the lower layers of the Basic Reference Model (ISO 7498), each user data parameter of a service primitive is specified as the binary value of a sequence of octets.

In the presentation layer, the nature of user data parameters changes. Application layer standards require the presentation service user data (see ISO 8822) to carry the value of quite complex types, possibly including strings of characters from a variety of character sets. In order to specify the value which is carried, they require a defined notation which does not determine the representation of the value. This is supplemented by the specification of one or more algorithms called **encoding rules** which determine the value of the session layer octets carrying such application layer values (called the **transfer syntax**). The presentation layer protocol (see ISO 8823) can negotiate which transfer syntaxes are to be used.

The purpose of specifying a value is to distinguish it from other possible values. The collection of the value together with the values from which it is distinguished is called a **type**, and one specific instance is a **value** of that type. More generally, a value or type can often be considered as composed of several simpler values or types, together with the relationships between them. The term **datatype** is often used as a synonym for type.

In order to correctly interpret the representation of a value (whether by marks on paper or bits on a communication line), it is necessary to know (usually from the context), the type of the value being represented. Thus the identification of a type is an important part of this International Standard.

A very general technique for defining a complicated type 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 a list 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 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) sequence or (unordered) set of zero, one or more values of the existing type; the (infinite) collection of all possible 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;

Types which are defined in this way are called **structured types**.

Every type defined using the notation specified in this International Standard is assigned a **tag**. The tag is defined either by this International Standard or by the user of the notation.

It is common for the same tag to be assigned to many different types, the particular type being identified by the context in which the tag is used.

The user of the notation may choose to assign distinct tags to two occurrences of a single type, thereby creating two distinct types. This can be necessary when it is required to distinguish which choice has been made in situations such as d) above.

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 International Standard, and each tag is either

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

The second class of tag is the **application** class. Application class tags are assigned to types by other standards. Within a particular standard, an application class tag is assigned to only one type.

The third class is the **private** class. Private class tags are never assigned by International Standards. Their use is enterprise specific.

The final class of tag is the **context-specific** class. This is freely assigned within any use of this notation, and is interpreted according to the context in which it is used.

Tags are mainly intended for machine use, and are not essential for the human notation defined in this 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.

NOTES

1 All types which can be defined in the notation of this International Standard have a tag. Given any type, the user of the notation can define a new type with a different tag.

2 Encoding rules always carry the tag of a type, explicitly or implicitly, with any representation of a value of the type. The restrictions placed on the use of the notation are designed to ensure that the tag is sufficient to unambiguously determine the actual type, provided the applicable type definitions are available.

This International Standard specifies a notation which both enables complicated types to be defined and also enables values of these types to

be specified. This is done without determining the way an instance of this type is to be represented (by a sequence of octets) during transfer. A notation which provides this facility is called a **notation for abstract syntax definition**.

The purpose of this International Standard is to specify a notation for abstract syntax definition called **Abstract Syntax Notation One**, or ASN.1. Abstract Syntax Notation One is used as a semi-formal tool to define protocols. The use of the notation does not necessarily preclude ambiguous specifications. It is the responsibility of the users of the notation to ensure that their specifications are not ambiguous.

This International Standard is supported by other standards which specify **encoding rules**. The application of encoding rules to the value of a type defined by ASN.1 results in a complete specification of the representation of values of that type during transfer (a transfer syntax).

This International Standard is technically aligned with the relevant parts of CCITT Recommendation X.409(1984).

Section one of this International Standard defines the simple types supported by ASN.1, and specifies the notation to be used for referencing simple types and defining structured types. Section one also specifies the notation to be used for specifying values of types defined using ASN.1.

Section two of this International Standard defines additional types (character string types) which, by the application of encoding rules for character sets, can be equated with the octetstring type.

Section three of this International Standard defines certain structured types which are considered to be of general utility, but which require no additional encoding rules.

Annex A is part of this International Standard, and specifies a notation for extending the basic ASN.1 notation. This is called the macro facility.

Annex B is part of this International Standard, and defines the object identifier tree for authorities supported by ISO.

Annex C is part of this International Standard and defines the object identifier tree for authorities supported by CCITT.

Annex D is part of this International Standard and defines the object identifier tree for joint use by ISO and CCITT.

BS 6962 : 1988

Annex E is not part of this International Standard, and provides examples and hints on the use of the ASN.1 notation.

Annex F is not part of this International Standard, and provides a summary of ASN.1 using the notation of clause 5.

An index of keywords, types and main terms is included at the end of this International Standard.

The text of this International Standard, and in particular the annexes B to D, are the subject of joint ISO-CCITT agreement.

1 Scope and field of application

This International Standard specifies a notation for abstract syntax definition called Abstract Syntax Notation One (ASN.1).

This 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.

This International Standard defines mechanisms for constructing new types from more basic types, and specifies a notation for defining such structured types and assigning them tags, and for specifying values of these types.

This International Standard defines character sets (by reference to other International Standards) for use within ASN.1.

This International Standard 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 protocol standards.

The ASN.1 notation is also referenced by other presentation layer standards which define encoding rules for the simple types, the structured types, the character string types and the useful types defined in ASN.1.

2 References

ISO 2014, *Writing of calendar dates in all-numeric form*.

ISO 2375, *Data processing - Procedure for registration of escape sequences*.

ISO 3166, *Codes for the representation of names of countries*.

ISO 3307, *Information interchange - Representations of time of the day*.

ISO 4031, *Information interchange - Representation of local time differentials*.

ISO 6523, *Data interchange - Structure for identification of organizations*.

ISO 7498, *Information processing systems - Open Systems Interconnection - Basic Reference Model (see also CCITT Recommendation X.200)*.

ISO 8822, *Information processing systems - Open Systems Interconnection - Connection-oriented presentation service definition*.

ISO 8823, *Information processing systems - Open Systems Interconnection - Connection-oriented presentation protocol specification*.

ISO 8825, *Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*.

CCITT X.409 (1984), *Message handling systems: presentation transfer syntax and notation*.

3 Definitions

The definitions in ISO 7498 are used in this International Standard.

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

3.2 type: A named set of values.

3.3 simple type: A type defined by directly specifying the set of its values.

3.4 structured type: A type defined by reference to one or more other types.

3.5 component type: One of the types referenced when defining a structured type.

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

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

3.8 ASN.1 character set: The set of characters, specified in clause 7, used in the ASN.1 notation.

3.9 items: Named sequences of characters from the ASN.1 character set, specified in clause 8, which are used to form the ASN.1 notation.

3.10 type (or value) reference name: A name associated uniquely with a type (or value) within some context.

NOTE — Reference names are assigned to the types defined in this International Standard; these are universally available within ASN.1. Other reference names are defined in other standards, and are applicable only in the context of that standard.

3.11 ASN.1 encoding rules: Rules which specify the representation during transfer of the value of any ASN.1 type; ASN.1 encoding rules enable information being transferred to be identified by the recipient as a specific value of a specific ASN.1 type.

3.12 character string type: A type whose values are strings of characters from some defined character set.

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

3.14 true: One of the distinguished values of the boolean type.

3.15 false: The other distinguished value of the boolean type.

3.16 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. Nonetheless, a particular user of ASN.1 may express a more restricted range by means of a comment.

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

NOTE — Encoding rules do not limit the number of bits in a bitstring. Nonetheless, a particular user of ASN.1 may express a minimum or maximum for the number of bits by means of a comment.

3.18 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.

NOTE — Encoding rules do not limit the number of octets in an octet string. Nonetheless, a particular user of ASN.1 may express a minimum or maximum for the number of octets by means of a comment.

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

NOTE — The null value is commonly used where several alternatives are possible, but none of them apply.

3.20 sequence type: A structured type, defined by referencing a fixed, ordered, list of types (some of which may be declared to be optional); each value of the new 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 new type need not contain a value of that component type.

3.21 sequence-of type: A structured type, defined by referencing a single existing type; each value in the new type is an ordered list of zero, one or more values of the existing type.

NOTE — Encoding rules do not limit the number of values in a sequence-of value. Nonetheless, a particular user of ASN.1 may express a minimum or maximum for the number of values by means of a comment.

3.22 set type: A structured type, defined by referencing a fixed, unordered, list of distinct types (some of which may be declared to be optional); each value in the new 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.

3.23 set-of type: A structured type, defined by referencing a single existing type; each value in the new type is an unordered list of zero, one or more values of the existing type.

NOTE — Encoding rules do not limit the number of values in a set-of value. Nonetheless, a particular user of ASN.1 may express a maximum or minimum for the number of values, or require each value to be distinct, by means of a comment.

3.24 tagged type: 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.

3.25 choice type: A structured type, defined by referencing a fixed, unordered, list of distinct types; each value of the new type is a value of one of the component types.

3.26 selection type: A structured type, defined by reference to a component type of a choice type.

3.27 any type: A choice type whose component types are unspecified, but are restricted to the set of types which can be defined using ASN.1.

3.28 external type: A type whose distinguished values cannot be deduced from their characterisation as external, but which can be deduced from the encoding of such a value; the values may, but need not, be describable using ASN.1, and thus their encodings may, but need not, conform to ASN.1 encoding rules.

3.29 information 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

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

3.31 object identifier type: A type whose distinguished values are the set of all object identifiers allocated in accordance with the rules of this International Standard.

NOTE — The rules of this International Standard permit a wide range of authorities to independently associate object identifiers with information objects.

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

NOTE — An object descriptor value is usually, but not always associated with a single information object. Only an object identifier value unambiguously identifies an information object.

3.33 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.

3.34 module: One or more instances of the use of the ASN.1 notation for type and value definition, encapsulated using the ASN.1 module notation (see clause 9).

3.35 production: A part of the formal notation used to specify ASN.1, in which allowed sequences of items are associated with a name which can be used to reference those sequences in the definition of new sets of allowed sequences.

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

NOTES

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.

2 UTC is also referred to as Greenwich Mean Time and appropriate time signals are regularly broadcast.

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

This International Standard uses the following terms defined in ISO 8822:

- a) presentation data value; and
- b) (an) abstract syntax; and
- c) abstract syntax name; and
- d) transfer syntax name.

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

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

4 Abbreviations

ASN.1 Abstract Syntax Notation One.

UTC Coordinated Universal Time.

ICD International Code Designator.

5 Notation used in this International Standard

The ASN.1 notation consists of a sequence of characters from the ASN.1 character set specified in clause 7.

Each use of the ASN.1 notation contains characters from the ASN.1 character set grouped into items. Clause 8 specifies all the sequences of characters forming ASN.1 items, and names each item.

The ASN.1 notation is specified in clause 9 (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 International Standard uses a formal notation defined in the following sub-clauses.

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 sequences defined in this International Standard and forms a new collection of sequences by specifying either

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

Each production consists of the following parts, in order:

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

::=

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

|

A 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 International Standard by the name in a) above.

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

5.2 The alternative collections

Each of the alternative collections of sequences in "one or more alternative collections of" is specified by a list of names. Each name is either the name of an item, or is the name of a collection of sequences defined by a production in this International Standard.

The collection of sequences defined by the alternative consists of all sequences obtained by taking any one of the sequences (or the item) associated with the first name, in combination with (and followed by) any one of the sequences (or item) associated with the second name, in combination with (and followed by) any one of the 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 sequences:

- any bstring (an item); and
- any hstring (an item); and
- any sequence associated with **IdentifierList**, preceded by a { and followed by a }

NOTE — { and } are the names of items containing the single characters { and } (see clause 8).

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 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 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.

Table 1 — Universal class tag assignments

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
UNIVERSAL 9 – 15	Reserved for addenda to this International Standard
UNIVERSAL 16	Sequence and Sequence – of types
UNIVERSAL 17	Set and Set – of types
UNIVERSAL 18 – 22, 25 – 27	Character string types
UNIVERSAL 23 – 24	Time types
UNIVERSAL 28 – ...	Reserved for addenda to this International Standard

5.6 References to a collection of sequences

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

5.7 References to an item

This International Standard references an item by referencing the name of the item; the name is surrounded by " to distinguish it from natural language text, unless it appears as part of a production.

5.8 Tags

A tag is specified by giving its class and the number within the class. The class is one of

universal
application
private
context-specific

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 24.

Tags in the universal class are assigned in such a way that, for structured types, the top-level structure can be deduced from the tag, and for

simple types, the type can be deduced from the tag. Table 1 summarises the assignment of tags in the universal class which are specified in this International Standard.

NOTE — Additional tags in the universal class are reserved for assignment by subsequent addenda to this International Standard.

6 Use of the ASN.1 notation

6.1 The ASN.1 notation for a type definition shall be "Type" (see 12.1).

6.2 The ASN.1 notation for a value of a type shall be "Value" (see 12.6).

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

6.3 The ASN.1 notation for assigning a type to a type reference name shall be "Typeassignment" (see 11.1).

6.4 The ASN.1 notation for assigning a value to a value reference name shall be "Valueassignment" (see 11.2).

6.5 The notation "Typeassignment" and "Valueassignment" shall only be used within the notation "ModuleDefinition" (but see 9.1).

Section one: Specification of ASN.1 notation

7 The ASN.1 character set

7.1 An ASN.1 item shall consist of a sequence of the characters listed in table 2, except as specified in 7.2 and 7.3.

Table 2 — ASN.1 characters

A to Z
a to z
0 to 9
: = , { } < .
() [] - ' "

NOTES

1 The additional characters > and | are used in the macro-notation (see annex A).

2 Where equivalent derivative standards are developed by national standards bodies, additional characters may appear in the following items (the last five of which are defined in annex A):

typereference	(8.2.1)
identifier	(8.3)
valuereference	(8.4)
modulereference	(8.5)
macrolreference	(A.2.1)
productionreference	(A.2.2)
localtypereference	(A.2.3)
localvaluereference	(A.2.4)
astring	(A.2.7)

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.

7.2 Where the notation is used to specify the value of a character string type, all characters of the defined character set can appear in the ASN.1 notation, surrounded by the characters ". (See 8.11).

7.3 Additional characters may appear in the "comment" item. (See 8.6).

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

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

8 ASN.1 items

8.1 General rules

8.1.1 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.

NOTE — Annex A specifies additional items used in the macro notation.

8.1.2 Each item specified in the following subclauses shall appear on a single line, and (except for the "comment" item) shall not contain spaces.

8.1.3 The length of a line is not restricted.

8.1.4 The items in the sequences specified by this International Standard (the ASN.1 notation) may appear on one line or may appear on several lines, and may be separated by one or more spaces or empty lines.

8.1.5 An item shall be separated from a following item by a space, or by being placed on a separate line, 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.

8.2 Type references

Name of item - typereference

8.2.1 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.

8.2.2 A "typereference" shall not be one of the reserved character sequences listed in table 3.

Table 3 — Reserved character sequences

BOOLEAN
INTEGER
BIT
STRING
OCTET
NULL
SEQUENCE
OF
SET
IMPLICIT
CHOICE
ANY
EXTERNAL
OBJECT
IDENTIFIER
OPTIONAL
DEFAULT
COMPONENTS
TRUE
FALSE
BEGIN
END

NOTE — A.2.9 specifies additional reserved character sequences when within a macro definition.

8.3 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.

8.4 Value references

Name of item - valuereference

A "valuereference" shall consist of the sequence of characters specified for an "identifier" in 8.3. In analysing an instance of use of this notation, a "valuereference" is distinguished from an "identifier" by the context in which it appears.

8.5 Module reference

Name of item - modulereference

A "modulereference" shall consist of the sequence of characters specified for a "typereference" in 8.2. In analysing an instance of use of this notation, a "modulereference" is distinguished from a "typereference" by the context in which it appears.

8.6 Comment

Name of item - comment

8.6.1 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 significance.

8.6.2 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 characters which are not in the character set specified in 7.1 (see 7.3).

8.7 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 sequences are specified, to indicate that absence of all alternatives is possible.

8.8 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.

8.9 Binary string item

Name of item - bstring

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

'B

EXAMPLE - '01101100'B

8.10 Hexadecimal string item

Name of item - hstring

8.10.1 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

preceded by a single ' and followed by the pair of characters

'H

EXAMPLE - 'AB0196'H

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

8.11 Character string item

Name of item - cstring

A "cstring" shall consist of an arbitrary number (possibly zero) of characters from the character set referenced by a character string type, preceded and followed by ". If the character set includes the character ", this character shall be represented in the "cstring" by a pair of ". The character set involved is not limited to the character set listed in table 2, but is determined by the type for which the "cstring" is a value (see 7.2).

EXAMPLE - "宛姐蛇飴絢"

8.12 Assignment item

Name of item - "::="

This item shall consist of the sequence of characters

::=

NOTE — This sequence does not contain any space characters (see 8.1.2).

8.13 Single character items

Names of items - {
}
<
,
.
(
)
[
]
- (hyphen)

An item with any of the names listed above shall consist of the single character forming the name.

8.14 Keyword items

Names of items - BOOLEAN
INTEGER
BIT
STRING
OCTET
NULL
SEQUENCE
OF
SET
IMPLICIT
CHOICE
ANY
EXTERNAL
OBJECT
IDENTIFIER
OPTIONAL
DEFAULT
COMPONENTS
UNIVERSAL
APPLICATION
PRIVATE
TRUE
FALSE
BEGIN
END
DEFINITIONS

Items with the above names shall consist of the sequence of characters in the name.

NOTES

1 Spaces do not occur in these sequences.

2 Where these sequences are not listed as reserved sequences in 8.2.2, they are distinguished from other items containing the same characters by the context in which they appear.

9 Module definition

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

ModuleDefinition ::= modulereference
DEFINITIONS
"::="

BEGIN
ModuleBody
END

ModuleBody ::= AssignmentList |
empty

AssignmentList ::=
Assignment |
AssignmentList Assignment

Assignment ::= Typeassignment |
Valueassignment

NOTES

1 Annex A specifies a "MacroDefinition" sequence which can also appear in the "AssignmentList". Notations defined by a macro definition may appear before or after the macro definition, within the same module.

2 In individual (but deprecated) cases, and for examples and for the definition of types with universal class tags, the "ModuleBody" can be used outside of a "ModuleDefinition".

3 "Typeassignment" and "Valueassignment" productions are specified in clause 11.

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

9.2 The "modulereference" appearing in the "ModuleDefinition" production is called the module name. Module names are chosen so as to ensure consistency and completeness of all "Assignment" sequences appearing within the "ModuleBody" of all "ModuleDefinition" sequences with this module name. A set of "Assignment" sequences is consistent and complete if, for every "typereference" or "valuereference" appearing within it, there is exactly one "Typeassignment" or "Valueassignment" (respectively) associating the name with a type or value (respectively).

9.3 Module names shall be used only once (except as specified in 9.4) within the sphere of interest of the definition of the module.

NOTE — It is recommended that modules defined in International Standards should have module names of the form

ISOxxxx-yyyy

where xxxx is the number of the International Standard, and yyyy is a suitable acronym for the International Standard (e.g. JTM, FTAM, or CCR). A similar convention can be applied in other standards.

9.4 A "typereference" or "valuereference" shall be referenced in a module different from that in which it is defined by using an "Externaltypereference" or "Externalvaluereference", specified by the following productions:

Externaltypereference ::=
modulereference
.
typereference

Externalvaluereference ::=
modulereference
.
valuereference

10 Referencing type and value definitions

10.1 The productions

DefinedType ::=
Externaltypereference |
typereference

DefinedValue ::=
Externalvaluereference |
valuereference

specify the sequences which shall be used, to reference type and value definitions.

10.2 The "typereference" and "valuereference" alternatives shall not be used unless the reference is within the module in which a type or value is assigned (see 11.1 and 11.2) to the typereference or valuereference.

10.3 The "Externaltypereference" and "Externalvaluereference" shall not be used unless the corresponding "typereference" or "valuereference" has been assigned a type or value respectively (see 11.1 and 11.2) within the corresponding "modulereference".

11 Assigning types and values

11.1 A "typereference" shall be assigned a type by the notation specified by the "Typeassignment" production:

Typeassignment ::= typereference
"::="
Type

The "typereference" shall not be one of the names used to reference the character string types defined in section two, and shall not be one of the names used to reference the types defined in section three.

11.2 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 valid notation (see 12.6) for a value of the type defined by "Type".

12 Definition of types and values

12.1 A type shall be referenced by one of the sequences "Type":

Type ::= BuiltinType | DefinedType
(see 10.1)

BuiltinType ::= BooleanType
IntegerType
BitStringType
OctetStringType
NullType
SequenceType
SequenceOfType
SetType
SetOfType
ChoiceType
SelectionType
TaggedType
AnyType
ObjectIdentifierType
CharacterStringType
UsefulType

NOTES

1 A type notation defined in a macro can also be used as a sequence for "Type" (see annex A).

2 Additional built-in types may be defined by addenda to this International Standard.

12.2 The "BuiltinType" notation is specified in the following clauses.

12.3 The type being referenced is the type defined by the "BuiltinType" or assigned to the "DefinedType".

12.4 In some notations within which a type is referenced, the type may be named. In such cases, this International Standard specifies the use of the notation "NamedType":

NamedType ::= identifier Type |
Type
SelectionType

The notation "SelectionType" and the corresponding value notation is specified in clause 23.

NOTE — The notation "SelectionType" contains an "identifier" which may form part of the value notation when "SelectionType" is used as a "NamedType" (see 23.1).

12.5 The "identifier" is not part of the type, and has no effect on the type. The type referenced by a "NamedType" sequence is that referenced by the contained "Type" sequence.

12.6 The value of a type shall be specified by one of the sequences "Value":

Value ::= BuiltinValue | DefinedValue

BuiltinValue ::= BooleanValue
IntegerValue
BitStringValue
OctetStringValue
NullValue
SequenceValue
SequenceOfValue
SetValue
SetOfValue
ChoiceValue
SelectionValue
TaggedValue
AnyValue
ObjectIdentifierValue
CharacterStringValue

NOTE — A value notation defined in a macro may also be used as a sequence for "Value" (see annex A).

12.7 If the type is defined using one of the notations shown on the left below, then the value shall be specified using the notation shown on the right below:

Type notation	Value notation
BooleanType	BooleanValue
IntegerType	IntegerValue
BitStringType	BitStringValue
OctetStringType	OctetStringValue
NullType	NullValue
SequenceType	SequenceValue
SequenceOfType	SequenceOfValue
SetType	SetValue
SetOfType	SetOfValue
ChoiceType	ChoiceValue
TaggedType	TaggedValue
AnyType	AnyValue
ObjectIdentifierType	ObjectIdentifierValue
CharacterStringType	CharacterStringValue

NOTE — Additional value notations may be defined by addenda to this International Standard.

Where the type is a DefinedType, the value notation shall be the notation for a type used in producing the DefinedType.

12.8 The value notation for a type defined by the "UsefulType" notation is specified in section three.

12.9 The "BuiltinValue" notation is specified in the following clauses.

12.10 The value of a type referenced using the "NamedType" notation shall be defined by the notation "NamedValue":

NamedValue ::= identifier Value |
Value

where the "identifier" (if any) is the same as that used in the "NamedType" notation. 23.2 specifies further restrictions on the "NamedValue" when the "NamedType" was a "SelectionType".

NOTE — The "identifier" is part of the notation, it does not form part of the value itself.

12.11 The "identifier" shall be present in the "NamedValue" if and only if it was present in the "NamedType".

NOTE — An "identifier" is always present in the case of a "SelectionType".

13 Notation for the boolean type

13.1 The boolean type (see 3.13) shall be referenced by the notation "BooleanType":

BooleanType ::= BOOLEAN

13.2 The tag for types defined by this notation is universal class, number 1.

13.3 The value of a boolean type (see 3.14 and 3.15) shall be defined by the notation "BooleanValue":

BooleanValue ::= TRUE | FALSE

14 Notation for the integer type

14.1 The integer type (see 3.16) shall be referenced by the notation "IntegerType":

IntegerType ::=
INTEGER |
INTEGER{NamedNumberList}

NamedNumberList ::=
NamedNumber |
NamedNumberList, NamedNumber

NamedNumber ::=
identifier(SignedNumber) |
identifier(DefinedValue)

SignedNumber ::= number | -number

14.2 The second alternative of "SignedNumber" shall not be used if the "number" is zero.

14.3 The "NamedNumberList" is not significant in the definition of a type. It is used solely in the value notation specified in 14.9

14.4 The "DefinedValue" shall be a reference to a value of type integer, or of a type derived from integer by tagging.

14.5 The value of each "SignedNumber" or "DefinedValue" appearing in the "NamedNumberList" shall be different, and represents a distinguished value of the integer type.

14.6 Each "identifier" appearing in the "NamedNumberList" shall be different.

14.7 The order of the "NamedNumber" sequences in the "NamedNumberList" is not significant.

14.8 The tag for types defined by this notation is universal class, number 2.

14.9 The value of an integer type shall be defined by the notation "IntegerValue":

IntegerValue ::= SignedNumber |
identifier

14.10 The "identifier" in "IntegerValue" shall be equal to that of an "identifier" in the "IntegerType" sequence with which the value is associated, and shall represent the corresponding number.

NOTE — When defining an integer value for which an "identifier" has been defined, use of the "identifier" form of "IntegerValue" should be preferred.

15 Notation for the bitstring type

15.1 The bitstring type (see 3.17) shall be referenced by the notation "BitStringType":

BitStringType ::=
BIT STRING |
BIT STRING{NamedBitList}

NamedBitList ::=
NamedBit |
NamedBitList, NamedBit

NamedBit ::=
identifier(number) |
identifier(DefinedValue)

15.2 The "NamedBitList" is not significant in the definition of a type. It is used solely in the value notation specified in 15.8.

15.3 The first bit in a bitstring has the number **zero**. The final bit in a bit string is called the **trailing bit**.