

Superseded by a more recent version



INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

X.690

(07/94)

**DATA NETWORKS AND OPEN SYSTEM
COMMUNICATIONS OSI NETWORKING
AND SYSTEM ASPECTS – ABSTRACT SYNTAX
NOTATION ONE (ASN.1)**

**INFORMATION TECHNOLOGY –
ASN.1 ENCODING RULES:
SPECIFICATION OF BASIC ENCODING
RULES (BER), CANONICAL ENCODING
RULES (CER) AND DISTINGUISHED
ENCODING RULES (DER)**

ITU-T Recommendation X.690

Superseded by a more recent version

(Previously “CCITT Recommendation”)

Superseded by a more recent version

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. Some 179 member countries, 84 telecom operating entities, 145 scientific and industrial organizations and 38 international organizations participate in ITU-T which is the body which sets world telecommunications standards (Recommendations).

The approval of Recommendations by the Members of ITU-T is covered by the procedure laid down in WTSC Resolution No. 1 (Helsinki, 1993). In addition, the World Telecommunication Standardization Conference (WTSC), which meets every four years, approves Recommendations submitted to it and establishes the study programme for the following period.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC. The text of ITU-T Recommendation X.690 was approved on 1st of July 1994. The identical text is also published as ISO/IEC International Standard 8825-1.

NOTE

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

© ITU 1995

All rights reserved. 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 the ITU.

Superseded by a more recent version

ITU-T X-SERIES RECOMMENDATIONS DATA NETWORKS AND OPEN SYSTEM COMMUNICATIONS

(February 1994)

ORGANIZATION OF X-SERIES RECOMMENDATIONS

Subject area	Recommendation Series
PUBLIC DATA NETWORKS	
Services and Facilities	X.1-X.19
Interfaces	X.20-X.49
Transmission, Signalling and Switching	X.50-X.89
Network Aspects	X.90-X.149
Maintenance	X.150-X.179
Administrative Arrangements	X.180-X.199
OPEN SYSTEMS INTERCONNECTION	
Model and Notation	X.200-X.209
Service Definitions	X.210-X.219
Connection-mode Protocol Specifications	X.220-X.229
Connectionless-mode Protocol Specifications	X.230-X.239
PICS Proformas	X.240-X.259
Protocol Identification	X.260-X.269
Security Protocols	X.270-X.279
Layer Managed Objects	X.280-X.289
Conformance Testing	X.290-X.299
INTERWORKING BETWEEN NETWORKS	
General	X.300-X.349
Mobile Data Transmission Systems	X.350-X.369
Management	X.370-X.399
MESSAGE HANDLING SYSTEMS	X.400-X.499
DIRECTORY	X.500-X.599
OSI NETWORKING AND SYSTEM ASPECTS	
Networking	X.600-X.649
Naming, Addressing and Registration	X.650-X.679
Abstract Syntax Notation One (ASN.1)	X.680-X.699
OSI MANAGEMENT	X.700-X.799
SECURITY	X.800-X.849
OSI APPLICATIONS	
Commitment, Concurrency and Recovery	X.850-X.859
Transaction Processing	X.860-X.879
Remote Operations	X.880-X.899
OPEN DISTRIBUTED PROCESSING	X.900-X.999

Superseded by a more recent version

CONTENTS

	<i>Page</i>
1 Scope.....	1
2 Normative references	1
2.1 Identical Recommendations International Standards	1
2.2 Additional references	1
3 Definitions.....	2
4 Abbreviations	2
5 Notation.....	2
6 Convention	3
7 Conformance	3
8 Basic encoding rules	3
8.1 General rules for encoding	3
8.2 Encoding of a boolean value	7
8.3 Encoding of an integer value	7
8.4 Encoding of an enumerated value	7
8.5 Encoding of a real value	8
8.6 Encoding of a bitstring value	9
8.7 Encoding of an octetstring value.....	10
8.8 Encoding of a null value	10
8.9 Encoding of a sequence value.....	11
8.10 Encoding of a sequence-of value	11
8.11 Encoding of a set value	11
8.12 Encoding of a set-of value	11
8.13 Encoding of a choice value	12
8.14 Encoding of a tagged value.....	12
8.15 Encoding of an open type	12
8.16 Encoding of an instance-of value.....	13
8.17 Encoding of a value of the embedded-pdv type.....	13
8.18 Encoding of a value of the external type.....	14
8.19 Encoding of an object identifier value	15
8.20 Encoding for values of the restricted character string types	16
8.21 Encoding for values of the unrestricted character string type	18
9 Canonical encoding rules	19
9.1 Length forms.....	19
9.2 String encoding forms.....	19
9.3 Set components	19
10 Distinguished encoding rules	19
10.1 Length forms.....	20
10.2 String encoding forms.....	20
10.3 Set components	20

Superseded by a more recent version

	<i>Page</i>
11	Restrictions on BER employed by both CER and DER..... 20
11.1	Boolean values 20
11.2	Unused bits 20
11.3	Real values 20
11.4	GeneralString values 21
11.5	Set and sequence components with default value 21
11.6	Set-of components 21
11.7	GeneralizedTime 21
12	Use of BER, CER and DER in transfer syntax definition 21
Annex A	– Example of encodings 23
A.1	ASN.1 description of the record structure 23
A.2	ASN.1 description of a record value 23
A.3	Representation of this record value 23
Annex B	– Assignment of object identifier values 25
Annex C	– Illustration of real value encoding 26
Annex D	– Use of the DER and CER in data origin authentication 28
D.1	The problem to be solved 28
D.2	The approach to a solution 29
D.3	The implementation optimization 29

Superseded by a more recent version

Summary

This Recommendation | International Standard defines a set of basic encoding rules (BER) that may be applied to values of types defined using the ASN.1 notation. Application of these encoding rules produces a transfer syntax for such values. It is implicit in the specification of these encoding rules that they are also used for decoding. This Recommendation | International Standard defines also a set of distinguished encoding rules (DER) and a set of canonical rules (CER) both of which provide constraints on the basic encoding rules (BER). The key difference between them is that DER uses the definite length form of encoding while CER uses the indefinite length form. DER is more suitable for the small encoded values, while CER is more suitable for the large ones. It is implicit in the specification of these encoding rules that they are also used for decoding.

Superseded by a more recent version

Introduction

ITU-T Rec. X.680 | ISO/IEC 8824-1, ITU-T Rec. X.681 | ISO/IEC 8824-2, ITU-T Rec. X.682 | ISO/IEC 8824-3, ITU-T Rec. X.683 | ISO/IEC 8824-4 (Abstract Syntax Notation One or ASN.1) together specify a notation for the definition of abstract syntaxes, enabling application layer standards to define the types of information they need to transfer using the presentation service. It also specifies a notation for the specification of values of a defined type.

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

There may be more than one set of encoding rules that can be applied to values of types that are defined using the ASN.1 notation. This Recommendation | International Standard defines three sets of encoding rules, called **basic encoding rules**, **canonical encoding rules** and **distinguished encoding rules**. Whereas the basic encoding rules gives the sender of an encoding various choices as to how data values may be encoded, the canonical and distinguished encoding rules select just one encoding from those allowed by the basic encoding rules, eliminating all of the sender's options. The canonical and distinguished encoding rules differ from each other in the set of restrictions that they place on the basic encoding rules.

The distinguished encoding rules is more suitable than the canonical encoding rules if the encoded value is small enough to fit into the available memory and there is a need to rapidly skip over some nested values. The canonical encoding rules is more suitable than the distinguished encoding rules if there is a need to encode values that are so large that they cannot readily fit into the available memory or it is necessary to encode and transmit a part of a value before the entire value is available. The basic encoding rules is more suitable than the canonical or distinguished encoding rules if the encoding contains a set value or set-of value and there is no need for the restrictions that the canonical and distinguished encoding rules impose. This is due to the memory and CPU overhead that the latter encoding rules exact in order to guarantee that set values and set-of values have just one possible encoding.

Annex A gives an example of the application of the basic encoding rules. It does not form an integral part of this Recommendation | International Standard.

Annex B summarizes the assignment of object identifier values made in this Recommendation | International Standard. It does not form an integral part of this Recommendation | International Standard.

Annex C gives examples of applying the basic encoding rules for encoding reals. It does not form an integral part of this Recommendation | International Standard.

Annex D provides a tutorial on the use of the distinguished encoding rules to provide an integrity service for OSI communications. It does not form an integral part of this Recommendation | International Standard.

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

**INFORMATION TECHNOLOGY – ASN.1 ENCODING RULES:
SPECIFICATION OF BASIC ENCODING RULES (BER),
CANONICAL ENCODING RULES (CER)
AND DISTINGUISHED ENCODING RULES (DER)**

1 Scope

This Recommendation | International Standard specifies a set of basic encoding rules that may be used to derive the specification of a transfer syntax for values of types defined using the notation specified in ITU-T Rec. X.680 (1994) | ISO/IEC 8824-1:1995, ITU-T Rec. X.681 (1994) | ISO/IEC 8824-2:1995, ITU-T Rec. X.682 (1994) | ISO/IEC 8824 3:1995, and ITU-T Rec. X.683 (1994) | ISO/IEC 8824-4:1995, collectively referred to as Abstract Syntax Notation One or ASN.1. These basic encoding rules are also to be applied for decoding such a transfer syntax in order to identify the data values being transferred. It also specifies a set of canonical and distinguished encoding rules that restrict the encoding of values to just one of the alternatives provided by the basic encoding rules.

These encoding rules are used at the time of communication (by the presentation service provider when required by a presentation context).

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 editions of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunications 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.226 (1994) | ISO/IEC 8823-1:1994, *Information technology – Open Systems Interconnection – Connection-oriented presentation protocol: Protocol specification.*
- ITU-T Recommendation X.680 (1994) | ISO/IEC 8824-1:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T Recommendation X.681 (1994) | ISO/IEC 8824-2:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification.*
- ITU-T Recommendation X.682 (1994) | ISO/IEC 8824-3:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification.*
- ITU-T Recommendation X.683 (1994) | ISO/IEC 8824-4:1995, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.*

2.2 Additional references

- ISO *International Register of Coded Character Sets to be used with Escape Sequence.*
- ISO/IEC 2022:1994, *Information processing – ISO 7-bit and 8-bit coded character sets – Code extension techniques.*
- ISO 6093:1985, *Information processing – Representation of numerical values in character strings for information interchange.*

Superseded by a more recent version ISO/IEC 8825-1 : 1995 (E)

- ISO 6429:1992, *Information technology – Control functions for coded character sets*.
- CCITT Recommendation X.208 (1988), *Specification of Abstract Syntax Notation One (ASN.1)*.
- ISO/IEC 8824-1 to 8824:1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*.
- ISO/IEC 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS): – Architecture and Basic Multilingual Plane*.

3 Definitions

For the purposes of this Recommendation | International Standard the definitions of ISO 7498 and ITU-T Rec. X.680 | ISO/IEC 8824-1 and the following definitions apply.

3.1 dynamic conformance: A statement of the requirement for an implementation to adhere to the behavior prescribed by this Recommendation | International Standard in an instance of communication.

3.2 static conformance: A statement of the requirement for support by an implementation of a valid set of features from among those defined by this Recommendation | International Standard.

3.3 data value: Information specified as the value of a type; the type and the value are defined using ASN.1.

3.4 encoding (of a data value): The complete sequence of octets used to represent the data value.

3.5 identifier octets: Part of a data value encoding which is used to identify the type of the value.

NOTE – Some ITU-T Recommendations use the term "data element" for this sequence of octets, but the term is not used in this Recommendation | International Standard, as other Recommendations | International Standards use it to mean "data value".

3.6 length octets: Part of a data value encoding following the identifier octets which is used to determine the end of the encoding.

3.7 contents octets: That part of a data value encoding which represents a particular value, to distinguish it from other values of the same type.

3.8 end-of-contents octets: Part of a data value encoding, occurring at its end, which is used to determine the end of the encoding.

NOTE – Not all encodings require end-of-contents octets.

3.9 primitive encoding: A data value encoding in which the contents octets directly represent the value.

3.10 constructed encoding: A data value encoding in which the contents octets are the complete encoding of one or more data values.

3.11 receiver: An implementation decoding the octets produced by a sender, in order to identify the data value which was encoded.

3.12 sender: An implementation encoding a data value for transfer.

3.13 trailing 0 bit: A 0 in the last position of a bitstring value.

NOTE – The 0 in a bitstring value consisting of a single 0 bit is a trailing 0 bit. Its removal produces an empty bitstring.

4 Abbreviations

ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules of ASN.1
CER	Canonical Encoding Rules of ASN.1
DER	Distinguished Encoding Rules of ASN.1
ULA	Upper Layer Architecture

5 Notation

This Recommendation | International Standard references the notation defined by ITU-T Rec. X.680 | ISO/IEC 8824-1.

6 Convention

6.1 This Recommendation | International Standard specifies the value of each octet in an encoding by use of the terms "most significant bit" and "least significant bit".

NOTE – Lower layer specifications use the same notation to define the order of bit transmission on a serial line, or the assignment of bits to parallel channels.

6.2 For the purposes of this Recommendation | International Standard only, the bits of an octet are numbered from 8 to 1, where bit 8 is the "most significant bit", and bit 1 is the "least significant bit".

6.3 For the purpose of this Recommendation | International Standard, two octet strings can be compared. One octet string is equal to another if they are of the same length and are the same at each octet position. An octet string, S_1 , is greater than another, S_2 , if and only if either:

- a) S_1 and S_2 have identical octets in every position up to and including the final octet in S_2 , but S_1 is longer; or
- b) S_1 and S_2 have different octets in one or more positions, and in the first such position, the octet in S_1 is greater than that in S_2 , considering the octets as unsigned binary numbers whose bit n has weight 2^{n-1} .

7 Conformance

7.1 Dynamic conformance is specified by clause to clause inclusive.

7.2 Static conformance is specified by those standards which specify the application of one or more of these encoding rules.

7.3 Alternative encodings are permitted by the basic encoding rules as a sender's option. Receivers who claim conformance to the basic encoding rules shall support all alternatives.

NOTE – Examples of such alternative encodings appear in 8.1.3.2 b) and Table 3.

7.4 No alternative encodings are permitted by the Canonical Encoding Rules or Distinguished Encoding Rules.

8 Basic encoding rules

8.1 General rules for encoding

8.1.1 Structure of an encoding

8.1.1.1 The encoding of a data value shall consist of four components which shall appear in the following order:

- a) identifier octets (see 8.1.2);
- b) length octets (see 8.1.3);
- c) contents octets (see 8.1.4);
- d) end-of-contents octets (see 8.1.5).

8.1.1.2 The end-of-contents octets shall not be present unless the value of the length octets requires them to be present (see 8.1.3).

8.1.1.3 Figure 1 illustrates the structure of an encoding (primitive or constructed). Figure 2 illustrates an alternative constructed encoding.

8.1.2 Identifier octets

8.1.2.1 The identifier octets shall encode the ASN.1 tag (class and number) of the type of the data value.

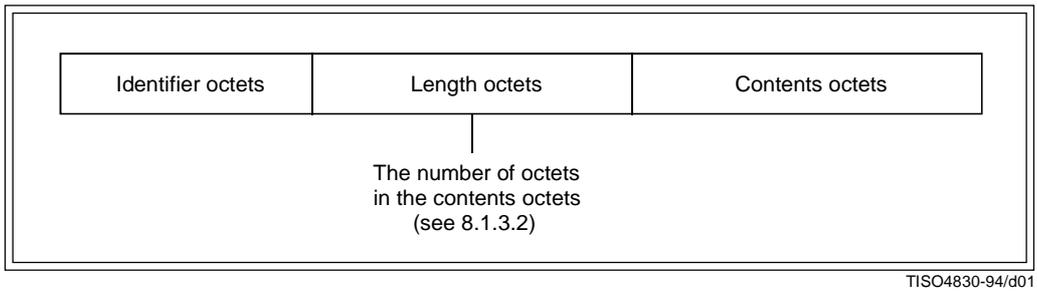


Figure 1 – Structure of an encoding

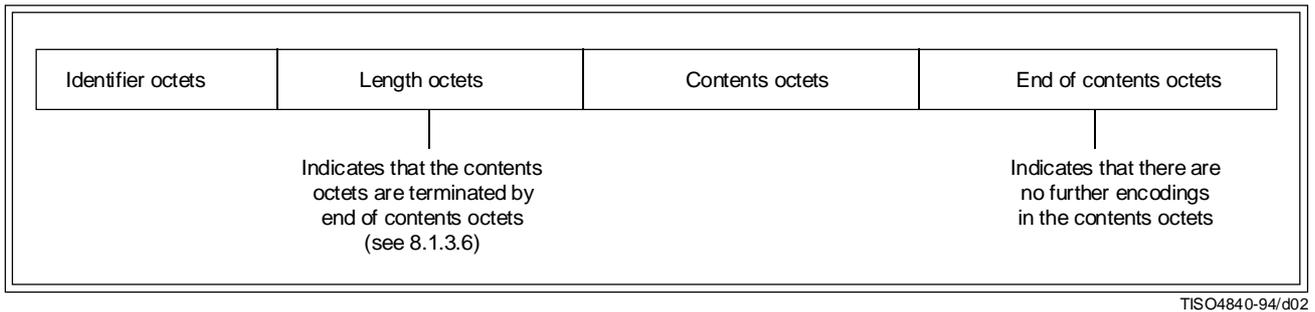


Figure 2 – An alternative constructed encoding

8.1.2.2 For tags with a number ranging from zero to 30 (inclusive), the identifier octets shall comprise a single octet encoded as follows:

- a) bits 8 and 7 shall be encoded to represent the class of the tag as specified in Table 1;
- b) bit 6 shall be a zero or a one according to the rules of 8.1.2.5;
- c) bits 5 to 1 shall encode the number of the tag as a binary integer with bit 5 as the most significant bit.

Table 1 – Encoding of class of tag

Class	Bit 8	Bit 7
Universal	0	0
Application	0	1
Context-specific	1	0
Private	1	1

8.1.2.3 Figure 3 illustrates the form of an identifier octet for a type with a tag whose number is in the range zero to 30 (inclusive).

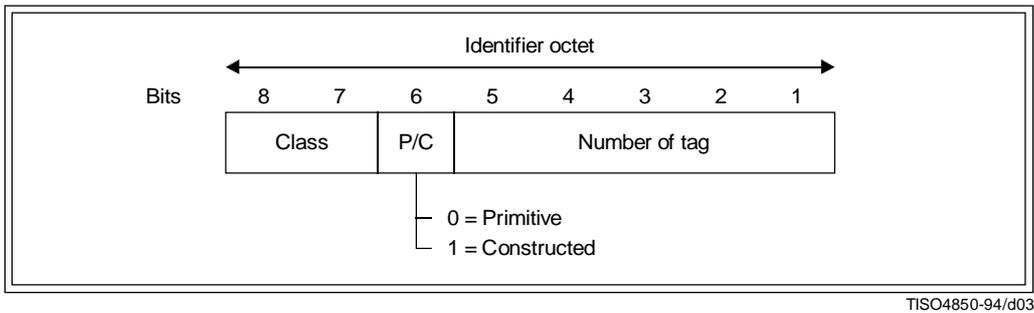


Figure 3 – Identifier octet (low tag number)

8.1.2.4 For tags with a number greater than or equal to 31, the identifier shall comprise a leading octet followed by one or more subsequent octets.

8.1.2.4.1 The leading octet shall be encoded as follows:

- a) bits 8 and 7 shall be encoded to represent the class of the tag as listed in Table 1;
- b) bit 6 shall be a zero or a one according to the rules of 8.1.2.5;
- c) bits 5 to 1 shall be encoded as 11111₂.

8.1.2.4.2 The subsequent octets shall encode the number of the tag as follows:

- a) bit 8 of each octet shall be set to one unless it is the last octet of the identifier octets;
- b) bits 7 to 1 of the first subsequent octet, followed by bits 7 to 1 of the second subsequent octet, followed in turn by bits 7 to 1 of each further octet, up to and including the last subsequent octet in the identifier octets shall be the encoding of an unsigned binary integer equal to the tag number, with bit 7 of the first subsequent octet as the most significant bit;
- c) bits 7 to 1 of the first subsequent octet shall not all be zero.

8.1.2.4.3 Figure 4 illustrates the form of the identifier octets for a type with a tag whose number is greater than 30.

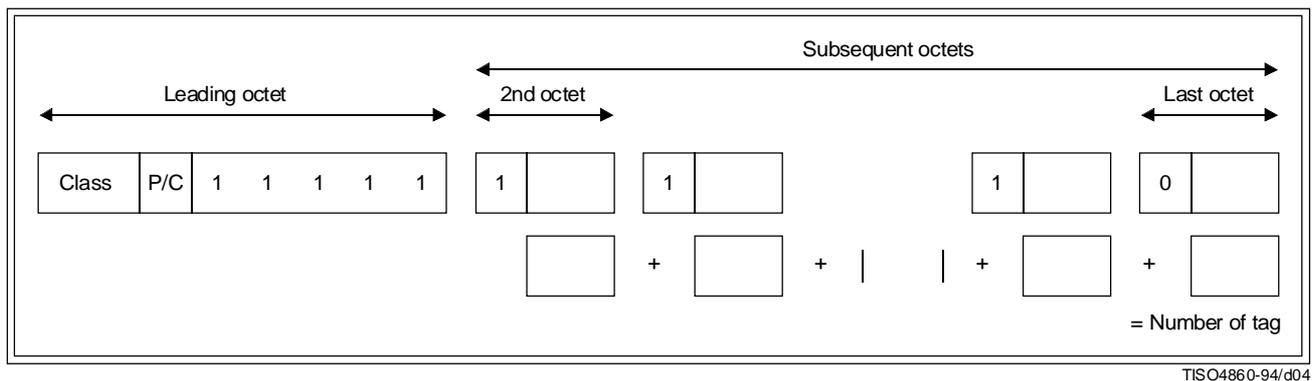


Figure 4 – Identifier octets (high tag number)

8.1.2.5 Bit 6 shall be set to zero if the encoding is primitive, and shall be set to one if the encoding is constructed.

NOTE – Subsequent clauses specify whether the encoding is primitive or constructed for each type.

8.1.2.6 ITU-T Rec. X.680 | ISO/IEC 8824-1 specifies that the tag of a type defined using the "CHOICE" keyword takes the value of the tag of the type from which the chosen data value is taken.

8.1.2.7 ITU-T Rec. X.681 | ISO/IEC 8824-2, subclauses 14.2 and 14.4 specifies that the tag of a type defined using "ObjectClassFieldType" is indeterminate if it is a type field, a variable-type value field, or a variable-type value set field. This type is subsequently defined to be an ASN.1 type, and the complete encoding is then identical to that of a value of the assigned type (including the identifier octets).

8.1.3 Length octets

8.1.3.1 Two forms of length octets are specified. These are:

- a) the definite form (see 8.1.3.3); and
- b) the indefinite form (see 8.1.3.6).

8.1.3.2 A sender shall:

- a) use the definite form (see 8.1.3.3) if the encoding is primitive;
- b) use either the definite form (see 8.1.3.3) or the indefinite form (see 8.1.3.6), a sender's option, if the encoding is constructed and all immediately available;
- c) use the indefinite form (see 8.1.3.6) if the encoding is constructed and is not all immediately available.

8.1.3.3 For the definite form, the length octets shall consist of one or more octets, and shall represent the number of octets in the contents octets using either the short form (see 8.1.3.4) or the long form (see 8.1.3.5) as a sender's option.

NOTE – The short form can only be used if the number of octets in the contents octets is less than or equal to 127.

8.1.3.4 In the short form, the length octets shall consist of a single octet in which bit 8 is zero and bits 7 to 1 encode the number of octets in the contents octets (which may be zero), as an unsigned binary integer with bit 7 as the most significant bit.

Example

L = 38 can be encoded as 00100110₂.

8.1.3.5 In the long form, the length octets shall consist of an initial octet and one or more subsequent octets. The initial octet shall be encoded as follows:

- a) bit 8 shall be one;
- b) bits 7 to 1 shall encode the number of subsequent octets in the length octets, as an unsigned binary integer with bit 7 as the most significant bit;
- c) the value 11111111₂ shall not be used.

NOTE 1 – This restriction is introduced for possible future extension.

Bits 8 to 1 of the first subsequent octet, followed by bits 8 to 1 of the second subsequent octet, followed in turn by bits 8 to 1 of each further octet up to and including the last subsequent octet, shall be the encoding of an unsigned binary integer equal to the number of octets in the contents octets, with bit 8 of the first subsequent octet as the most significant bit.

Example

L = 201 can be encoded as:

10000001₂

11001001₂

NOTE 2 – In the long form, it is a sender's option whether to use more length octets than the minimum necessary.

8.1.3.6 For the indefinite form, the length octets indicate that the contents octets are terminated by end-of-contents octets (see 8.1.5), and shall consist of a single octet.

8.1.3.6.1 The single octet shall have bit 8 set to one, and bits 7 to 1 set to zero.

8.1.3.6.2 If this form of length is used, then end-of-contents octets (see 8.1.5) shall be present in the encoding following the contents octets.

8.1.4 Contents octets

The contents octets shall consist of zero, one or more octets, and shall encode the data value as specified in subsequent clauses.

NOTE – The contents octets depend on the type of the data value; subsequent clauses follow the same sequence as the definition of types in ASN.1.

8.1.5 End-of-contents octets

The end-of-contents octets shall be present if the length is encoded as specified in 8.1.3.6, otherwise they shall not be present.

The end-of-contents octets shall consist of two zero octets.

NOTE – The end-of-contents octets can be considered as the encoding of a value whose tag is universal class, whose form is primitive, whose number of the tag is zero, and whose contents are absent, thus:

End-of-contents	Length	Contents
00 ₁₆	00 ₁₆	Absent

8.2 Encoding of a boolean value

8.2.1 The encoding of a boolean value shall be primitive. The contents octets shall consist of a single octet.

8.2.2 If the boolean value is
FALSE

the octet shall be zero.

If the boolean value is

TRUE

the octet shall have any non-zero value, as a sender's option.

Example – If of type BOOLEAN, the value TRUE can be encoded as:

Boolean	Length	Contents
01 ₁₆	01 ₁₆	FF ₁₆

8.3 Encoding of an integer value

8.3.1 The encoding of an integer value shall be primitive. The contents octets shall consist of one or more octets.

8.3.2 If the contents octets of an integer value encoding consist of more than one octet, then the bits of the first octet and bit 8 of the second octet

- a) shall not all be ones; and
- b) shall not all be zero.

NOTE – These rules ensure that an integer value is always encoded in the smallest possible number of octets.

8.3.3 The contents octets shall be a two's complement binary number equal to the integer value, and consisting of bits 8 to 1 of the first octet, followed by bits 8 to 1 of the second octet, followed by bits 8 to 1 of each octet in turn up to and including the last octet of the contents octets.

NOTE – The value of a two's complement binary number is derived by numbering the bits in the contents octets, starting with bit 1 of the last octet as bit zero and ending the numbering with bit 8 of the first octet. Each bit is assigned a numerical value of 2^N , where N is its position in the above numbering sequence. The value of the two's complement binary number is obtained by summing the numerical values assigned to each bit for those bits which are set to one, excluding bit 8 of the first octet, and then reducing this value by the numerical value assigned to bit 8 of the first octet if that bit is set to one.

8.4 Encoding of an enumerated value

The encoding of an enumerated value shall be that of the integer value with which it is associated.

NOTE – It is primitive.

8.5 Encoding of a real value

8.5.1 The encoding of a real value shall be primitive.

8.5.2 If the real value is the value zero, there shall be no contents octets in the encoding.

8.5.3 If the real value is non-zero, then the base used for the encoding shall be B' , chosen by the sender. If B' is 2, 8 or 16, a binary encoding, specified in 8.5.5, shall be used. If B' is 10, a character encoding, specified in 8.5.6, shall be used.

NOTE – The form of storage, generation, or processing by senders and receivers, and the form used in the ASN.1 value notation are all independent of the base used for transfer.

8.5.4 Bit 8 of the first contents octet shall be set as follows:

- a) if bit 8 = 1, then the binary encoding specified in 8.5.5 applies;
- b) if bit 8 = 0 and bit 7 = 0, then the decimal encoding specified in 8.5.6 applies;
- c) if bit 8 = 0 and bit 7 = 1, then a "SpecialRealValue" (see ITU-T Rec. X.680 | ISO/IEC 8824-1) is encoded as specified in 8.5.7.

8.5.5 When binary encoding is used (bit 8 = 1), then if the mantissa, M is non-zero, it shall be represented by a sign S , a non-negative integer value N and a binary scaling factor F , such that

$$M = S \times N \times 2^F$$

$$0 \leq F < 4$$

$$S = +1 \text{ or } -1$$

NOTE – The binary scaling factor F is required under certain circumstances in order to align the implied point of the mantissa to the position required by the encoding rules of this clause. This alignment can not always be achieved by modification of the exponent E . If the base B' used for encoding is 8 or 16, the implied point can only be moved in steps of 3 or 4 bits, respectively, by changing the component E . Therefore, values of the binary scaling factor F other than zero may be required in order to move the implied point to the required position.

8.5.5.1 Bit 7 of the first contents octets shall be 1 if S is -1 and 0 otherwise.

8.5.5.2 Bits 6 to 5 of the first contents octets shall encode the value of the base B' as follows:

<i>Bits 6 to 5</i>	<i>Base</i>
00	base 2
01	base 8
10	base 16
11	Reserved for further editions of this Recommendation International Standard.

8.5.5.3 Bits 4 to 3 of the first contents octet shall encode the value of the binary scaling factor F as an unsigned binary integer.

8.5.5.4 Bits 2 to 1 of the first contents octet shall encode the format of the exponent as follows:

- a) if bits 2 to 1 are 00, then the second contents octet encodes the value of the exponent as a two's complement binary number;
- b) if bits 2 to 1 are 01, then the second and third contents octets encode the value of the exponent as a two's complement binary number;
- c) if bits 2 to 1 are 10, then the second, third and fourth contents octets encode the value of the exponent as a two's complement binary number;
- d) if bits 2 to 1 are 11, then the second contents octet encodes the number of octets, X say, (as an unsigned binary number) used to encode the value of the exponent, and the third up to the $(X + 3)^{\text{th}}$ (inclusive) contents octets encode the value of the exponent as a two's complement binary number; the value of X shall be at least one; the first nine bits of the transmitted exponent shall not be all zeros or all ones.

8.5.5.5 The remaining contents octets encode the value of the integer N (see 8.5.5) as an unsigned binary number.

NOTES

1 For non-canonical BER there is no requirement for floating point normalization of the mantissa. This allows an implementor to transmit octets containing the mantissa without performing shift functions on the mantissa in memory. In the Canonical Encoding Rules and the Distinguished Encoding Rules normalization is specified and the mantissa (unless it is 0) needs to be repeatedly shifted until the least significant bit is a 1.

2 This representation of real numbers is very different from the formats normally used in floating point hardware, but has been designed to be easily converted to and from such formats (see annex C).

8.5.6 When decimal encoding is used (bits 8 to 7 = 00), all the contents octets following the first contents octet form a field, as the term is used in ISO 6093, of a length chosen by the sender, and encoded according to ISO 6093. The choice of ISO 6093 number representation is specified by bits 6 to 1 of the first contents octet as follows:

<i>Bits 6 to 1</i>	<i>Number representation</i>
00 0001	ISO 6093 NR1 form
00 0010	ISO 6093 NR2 form
00 0011	ISO 6093 NR3 form

The remaining values of bits 6 to 1 are reserved for to this Recommendation | International Standard.

There shall be no use of scaling factors specified in accompanying documentation (see ISO 6093).

NOTES

1 The recommendations in ISO 6093 concerning the use of at least one digit to the left of the decimal mark are also recommended in this Recommendation | International Standard, but are not mandatory.

2 Use of the normalized form (see ISO 6093) is a sender's option, and has no significance.

8.5.7 When "SpecialRealValues" are to be encoded (bits 8 to 7 = 01), there shall be only one contents octet, with values as follows:

01000000	Value is PLUS-INFINITY
01000001	Value is MINUS-INFINITY

All other values having bits 8 and 7 equal to 0 and 1 respectively are reserved for addenda to this Recommendation | International Standard.

8.6 Encoding of a bitstring value

8.6.1 The encoding of a bitstring value shall be either primitive or constructed at the option of the sender.

NOTE – Where it is necessary to transfer part of a bit string before the entire bitstring is available, the constructed encoding is used.

8.6.2 The contents octets for the primitive encoding shall contain an initial octet followed by zero, one or more subsequent octets.

8.6.2.1 The bits in the bitstring, commencing with the first bit and proceeding to the last bit, shall be placed in bits 8 to 1 of the first subsequent octet, followed by bits 8 to 1 of the second subsequent octet, followed by bits 8 to 1 of each octet in turn, followed by as many bits as are needed of the final subsequent octet, commencing with bit 8.

NOTE – The terms "first bit" and "trailing bit" are defined in ITU-T Rec. X.680 | ISO/IEC 8824-1.

8.6.2.2 The initial octet shall encode, as an unsigned binary integer with bit 1 as the least significant bit, the number of unused bits in the final subsequent octet. The number shall be in the range zero to seven.

8.6.2.3 If the bitstring is empty, there shall be no subsequent octets, and the initial octet shall be zero.

8.6.2.4 Where ITU-T Rec. X.680 | ISO/IEC 8824-1, subclause 19.7 applies, a BER encoder/decoder can add or remove trailing 0 bits from the value.

NOTE – If a bitstring value has no 1 bits then an encoder (as a sender's option) may encode the value with an initial octet set to 0 or may encode it as a bit string with one or more 0 bits following the initial octet.

8.6.3 The contents octets for the constructed encoding shall consist of zero, one, or more nested encodings.

NOTE – Each such encoding includes identifier, length, and contents octets, and may include end-of-contents octets if it is constructed.

8.6.4 To encode a bitstring value in this way, it is segmented. Each segment shall consist of a series of consecutive bits of the value, and with the possible exception of the last, shall contain a number of bits which is a multiple of eight. Each bit in the overall value shall be in precisely one segment, but there shall be no significance placed on the segment boundaries.

NOTE – A segment may be of size zero, i.e. contain no bits.

8.6.4.1 Each encoding in the contents octets shall represent a segment of the overall bitstring, the encoding arising from a recursive application of this clause. In this recursive application, each segment is treated as if it were a bitstring value. The encodings of the segments shall appear in the contents octets in the order in which their bits appear in the overall value.

NOTES

1 As a consequence of this recursion, each encoding in the contents octets may itself be primitive or constructed. However, such encodings will usually be primitive.

2 In particular, the tags in the contents octets are always universal class, number 3.

8.6.4.2 Example – If of type BIT STRING, the value '0A3B5F291CD'H can be encoded as shown below. In this example, the Bit String is represented as a primitive:

BitString	Length	Contents
03 ₁₆	07 ₁₆	040A3B5F291CD0 ₁₆

The value shown above can also be encoded as shown below. In this example, the Bit String is represented as a constructor:

BitString	Length	Contents		
23 ₁₆	80 ₁₆			
		BitString	Length	Contents
		03 ₁₆	03 ₁₆	000A3B ₁₆
		03 ₁₆	05 ₁₆	045F291CD0 ₁₆
EOC	Length			
00 ₁₆	00 ₁₆			

8.7 Encoding of an octetstring value

8.7.1 The encoding of an octetstring value shall be either primitive or constructed at the option of the sender.

NOTE – Where it is necessary to transfer part of an octet string before the entire octetstring is available, the constructed encoding is used.

8.7.2 The primitive encoding contains zero, one or more contents octets equal in value to the octets in the data value, in the order they appear in the data value, and with the most significant bit of an octet of the data value aligned with the most significant bit of an octet of the contents octets.

8.7.3 The contents octets for the constructed encoding shall consist of zero, one, or more encodings.

NOTE – Each such encoding includes identifier, length, and contents octets, and may include end-of-contents octets if it is constructed.

8.7.3.1 To encode an octetstring value in this way, it is segmented. Each segment shall consist of a series of consecutive octets of the value. There shall be no significance placed on the segment boundaries.

NOTE – A segment may be of size zero, i.e. contain no octets.

8.7.3.2 Each encoding in the contents octets shall represent a segment of the overall octetstring, the encoding arising from a recursive application of this subclause. In this recursive application, each segment is treated as if it were a octetstring value. The encodings of the segments shall appear in the contents octets in the order in which their octets appear in the overall value.

NOTES

1 As a consequence of this recursion, each encoding in the contents octets may itself be primitive or constructed. However, such encodings will usually be primitive.

2 In particular, the tags in the contents octets are always universal class, number 4.

8.8 Encoding of a null value

8.8.1 The encoding of a null value shall be primitive.

8.8.2 The contents octets shall not contain any octets.

NOTE – The length octet is zero.

Example – If of type NULL, the NULL can be encoded as:

<i>Null</i>	<i>Length</i>
05 ₁₆	00 ₁₆

8.9 Encoding of a sequence value

8.9.1 The encoding of a sequence value shall be constructed.

8.9.2 The contents octets shall consist of the complete encoding of one data value from each of the types listed in the ASN.1 definition of the sequence type, in the order of their appearance in the definition, unless the type was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT".

8.9.3 The encoding of a data value may, but need not, be present for a type which was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT". If present, it shall appear in the encoding at the point corresponding to the appearance of the type in the ASN.1 definition.

Example – If of type

SEQUENCE {name IA5String, ok BOOLEAN}

the value

{name "Smith", ok TRUE}

can be encoded as:

Sequence	Length	Contents		
30 ₁₆	0A ₁₆	IA5String	Length	Contents
		16 ₁₆	05 ₁₆	"Smith"
		Boolean	Length	Contents
		01 ₁₆	01 ₁₆	FF ₁₆

8.10 Encoding of a sequence-of value

8.10.1 The encoding of a sequence-of value shall be constructed.

8.10.2 The contents octets shall consist of zero, one or more complete encodings of data values from the type listed in the ASN.1 definition.

8.10.3 The order of the encodings of the data values shall be the same as the order of the data values in the sequence-of value to be encoded.

8.11 Encoding of a set value

8.11.1 The encoding of a set value shall be constructed.

8.11.2 The contents octets shall consist of the complete encoding of a data value from each of the types listed in the ASN.1 definition of the set type, in an order chosen by the sender, unless the type was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT".

8.11.3 The encoding of a data value may, but need not, be present for a type which was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT".

NOTE – The order of data values in a set value is not significant, and places no constraints on the order during transfer.

8.12 Encoding of a set-of value

8.12.1 The encoding of a set-of value shall be constructed.

8.12.2 The text of 8.10.2 applies.

8.12.3 The order of data values need not be preserved by the encoding and subsequent decoding.

8.13 Encoding of a choice value

The encoding of a choice value shall be the same as the encoding of a value of the chosen type.

NOTES

- 1 The encoding may be primitive or constructed depending on the chosen type.
- 2 The tag used in the identifier octets is the tag of the chosen type, as specified in the ASN.1 definition of the choice type.

8.14 Encoding of a tagged value

8.14.1 The encoding of a tagged value shall be derived from the complete encoding of the corresponding data value of the type appearing in the "TaggedType" notation (called the base encoding) as specified in 8.14.2 and 8.14.3.

8.14.2 If implicit tagging (see ITU-T Rec. X.680 | ISO/IEC 8824-1, subclause 28.6) was not used in the definition of the type, the encoding shall be constructed and the contents octets shall be the complete base encoding.

8.14.3 If implicit tagging was used in the definition of the type, then:

- a) the encoding shall be constructed if the base encoding is constructed, and shall be primitive otherwise; and
- b) the contents octets shall be the same as the contents octets of the base encoding.

Example – With ASN.1 type definitions (in an explicit tagging environment) of

```

Type1 ::= VisibleString
Type2 ::= [APPLICATION 3] IMPLICIT Type1
Type3 ::= [2] Type2
Type4 ::= [APPLICATION 7] IMPLICIT Type3
Type5 ::= [2] IMPLICIT Type2
    
```

a value of

"Jones"

is encoded as follows:

For Type1:

VisibleString	Length	Contents
1A ₁₆	05 ₁₆	4A6F6E6573 ₁₆

For Type2:

[Application 3]	Length	Contents
43 ₁₆	05 ₁₆	4A6F6E6573 ₁₆

For Type3:

[2]	Length	Contents
A2 ₁₆	07 ₁₆	[Application 3]
		43 ₁₆
		Length
		05 ₁₆
		Contents
		4A6F6E6573 ₁₆

For Type4:

[Application 7]	Length	Contents
67 ₁₆	07 ₁₆	[Application 3]
		43 ₁₆
		Length
		05 ₁₆
		Contents
		4A6F6E6573 ₁₆

For Type5:

[2]	Length	Contents
82 ₁₆	05 ₁₆	4A6F6E6573 ₁₆

8.15 Encoding of an open type

The value of an open type is also a value of some (other) ASN.1 type. The encoding of such a value shall be the complete encoding herein specified for the value considered as being of that other type.

8.16 Encoding of an instance-of value

8.16.1 The encoding of the instance-of type shall be the BER encoding of the following sequence type with the value as specified in 8.16.2:

```
[UNIVERSAL 8] IMPLICIT SEQUENCE
{
  type-id  <DefinedObjectClass>.&id,
  value    [0] EXPLICIT <DefinedObjectClass>.&Type
}
```

where "<DefinedObjectClass>" is replaced by the particular "DefinedObjectClass" used in the "InstanceOfType" notation.

NOTE – When the value is a value of a single ASN.1 type and BER encoding is used for it, the encoding of this type is identical to an encoding of a corresponding value of the external type, where the "syntax" alternative is in use for representing the abstract value.

8.16.2 The value of the components of the sequence type in 8.16.1 shall be the same as the values of the corresponding components of the associated type in ITU-T Rec. X.681 | ISO/IEC 8824-2, subclause C.7.

8.17 Encoding of a value of the embedded-pdv type

8.17.1 There are two sub-rules used for encoding the embedded-pdv type, called EP-A (the index-setting sub-rule) and EP-B (the index-using sub-rule). For any given value of "identification" in the abstract value, the first occurrence (within the encoding of an entire pdv) of a value of the embedded-pdv type with this value of "identification" shall be encoded using sub-rule EP-A.

8.17.2 Subject to certain restrictions listed below, subsequent values with the same value of "identification" shall be encoded using sub-rule EP-B with the "index" set to the same value as the "index" in the corresponding EP-A encoding.

8.17.3 The conditions for use of sub-rule EP-B are listed below. If any condition is not satisfied, then sub-rule EP-A shall be used instead. The conditions for use of sub-rule EP-B are:

- a) the "index" value is in the range 0 to 255;
- b) the encoding of this instance of a value of the embedded-pdv type is an integral multiple of eight bits;
- c) the length of the encoding is not greater than the maximum length that can be identified by a long-form length encoding.

NOTE – The last condition is not likely to prove a restriction in practice.

8.17.4 For the first occurrence of an EP-A encoding the "index" shall have the value zero, and for each subsequent occurrence it shall be incremented by one.

8.17.5 TUTORIAL – Thus, for any given "identification" value there is a (relatively inefficient) EP-A encoding with both a unique index value and the full "identification" value, followed by arbitrarily many (efficient) EP-B encodings linked to the EP-A encoding by the index value. Because the EP-B encoding uses a single octet for the index, and a count in octets for the encoding, it cannot be used if the index value exceeds 255 (256 different "identifications" are in use), or if the encoding is not a multiple of eight bits. In these cases the EP-A encoding is used for all such occurrences.

8.17.6 The EP-A encoding shall be the BER encoding of the following sequence type after the application of the AUTOMATIC TAGS as specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, subclauses 22.6 and 26.3:

```
[UNIVERSAL 11] IMPLICIT SEQUENCE {
  index          INTEGER,
  identification CHOICE {
    syntaxes     SEQUENCE {
      abstract    OBJECT IDENTIFIER,
      transfer    OBJECT IDENTIFIER },
    syntax       OBJECT IDENTIFIER,
    presentation-context-id INTEGER,
    context-negotiation SEQUENCE {
      presentation-context-id INTEGER,
      transfer-syntax    OBJECT IDENTIFIER },
    transfer-syntax OBJECT IDENTIFIER,
    fixed         NULL },
  data-value     BIT STRING }
```

8.17.7 The value of "data-value" shall be the encoding of the abstract data value using the identified transfer syntax, the value of "index" shall be as determined above, and the value of all other fields shall be the same as the values appearing in the abstract value.

NOTES

1 The component "index" is not defined in the abstract syntax because this is meant to be supplied strictly at the encoding rule level (in the same sense that end-of-contents octets are meant to be applied at the encoding rule level).

2 Both alternatives of "data-value" in the abstract syntax are encoded identically – as a bitstring – in the transfer syntax.

8.17.8 The EP-B encoding shall be the BER encoding of the following ASN.1 type:

[UNIVERSAL 11] IMPLICIT OCTET STRING

where

- a) the encoding is primitive;
- b) the length octets is the short or long definite form as a sender's option;
- c) the contents octets encode the "index" in the first octet of the octet string value as an integer that ranges in value from 0 through 255, followed by the encoding of the "data-value".

NOTE – A receiver can distinguish the EP-A encoding from the EP-B encoding by the setting of the primitive/constructor bit.

8.18 Encoding of a value of the external type

8.18.1 The encoding of a value of the external type shall be the BER encoding of the following sequence type, assumed to be defined in an environment of EXPLICIT TAGS, with a value as specified in the subclauses below:

```
[UNIVERSAL 8] IMPLICIT SEQUENCE {
    direct-reference          OBJECT IDENTIFIER OPTIONAL,
    indirect-reference       INTEGER OPTIONAL,
    data-value-descriptor   ObjectDescriptor OPTIONAL,
    encoding                 CHOICE {
        single-ASN1-type    [0] ABSTRACT-SYNTAX.&Type,
        octet-aligned       [1] IMPLICIT OCTET STRING,
        arbitrary           [2] IMPLICIT BIT STRING } }
```

NOTE – This sequence type is the same as that specified in CCITT Rec. X.208 (1988) | ISO/IEC 8824 (1990), and the resulting encoding of a value of an external type is unchanged from those specifications.

8.18.2 The value of the fields depends on the abstract values being transmitted, which is a value of the type specified in 30.5 of ITU-T Rec. X.680 | ISO/IEC 8824-1.

8.18.3 The "data-value-descriptor" above shall be present if and only if the "data-value-descriptor" is present in the abstract value, and shall have the same value.

8.18.4 Values of "direct-reference" and "indirect-reference" above shall be present or absent in accordance with Table 2. Table 2 maps the external type alternatives of "identification" defined in ITU-T Rec. X.680 | ISO/IEC 8824-1, subclause 30.5 to the external type components "direct-reference" and "indirect-reference" defined in.

Table 2 – Alternative encodings for "identification"

identification	direct-reference	indirect-reference
syntaxes	*** CANNOT OCCUR ***	*** CANNOT OCCUR ***
syntax	syntax	ABSENT
presentation-context-id	ABSENT	presentation-context-id
context-negotiation	transfer-syntax	presentation-context-id
transfer-syntax	*** CANNOT OCCUR ***	*** CANNOT OCCUR ***
fixed	*** CANNOT OCCUR ***	*** CANNOT OCCUR ***

8.18.5 The data value shall be encoded according to the transfer syntax identified by the encoding, and shall be placed in an alternative of the "encoding" choice as specified below.

8.18.6 If the data value is the value of a single ASN.1 data type, and if the encoding rules for this data value are the same as those for the complete "EXTERNAL" data type, then the sending implementation shall use any of the "Encoding" choices:

- single-ASN1-type
- octet-aligned
- arbitrary

as an implementation option.

8.18.7 If the encoding of the data value, using the agreed or negotiated encoding, is an integral number of octets, then the sending implementation shall use any of the "Encoding" choices:

- octet-aligned,
- arbitrary,

as an implementation option.

NOTE – A data value which is a series of ASN.1 types, and for which the transfer syntax specifies simple concatenation of the octet strings produced by applying the ASN.1 Basic Encoding Rules to each ASN.1 type, falls into this category, not that of.

8.18.8 If the encoding of the data value, using the agreed or negotiated encoding, is not an integral number of octets, the "Encoding" choice shall be

- arbitrary

8.18.9 If the "Encoding" choice is chosen as "single-ASN1-type", then the ASN.1 type shall replace the open type, with a value equal to the data value to be encoded.

NOTE – The range of values which might occur in the open type is determined by the registration of the object identifier value associated with the "direct-reference", and/or the integer value associated with the "indirect-reference".

8.18.10 If the "Encoding" choice is chosen as "octet-aligned", then the data value shall be encoded according to the agreed or negotiated transfer syntax, and the resulting octets shall form the value of the octetstring.

8.18.11 If the "Encoding" choice is chosen as "arbitrary", then the data value shall be encoded according to the agreed or negotiated transfer syntax, and the result shall form the value of the bitstring.

8.19 Encoding of an object identifier value

8.19.1 The encoding of an object identifier value shall be primitive.

8.19.2 The contents octets shall be an (ordered) list of encodings of subidentifiers (see 8.19.3 and 8.19.4) concatenated together.

Each subidentifier is represented as a series of (one or more) octets. Bit 8 of each octet indicates whether it is the last in the series: bit 8 of the last octet is zero; bit 8 of each preceding octet is one. Bits 7-1 of the octets in the series collectively encode the subidentifier. Conceptually, these groups of bits are concatenated to form an unsigned binary number whose most significant bit is bit 7 of the first octet and whose least significant bit is bit 1 of the last octet. The subidentifier shall be encoded in the fewest possible octets, that is, the leading octet of the subidentifier shall not have the value 80₁₆.

8.19.3 The number of subidentifiers (N) shall be one less than the number of object identifier components in the object identifier value being encoded.

8.19.4 The numerical value of the first subidentifier is derived from the values of the first **two** object identifier components in the object identifier value being encoded, using the formula

$$(X*40) + Y$$

where X is the value of the first object identifier component and Y is the value of the second object identifier component.

NOTE – This packing of the first two object identifier components recognizes that only three values are allocated from the root node, and at most 39 subsequent values from nodes reached by $X = 0$ and $X = 1$.

8.19.5 The numerical value of the i 'th subidentifier, ($2 \leq i \leq N$) is that of the $(i + 1)$ 'th object identifier component.

Example – An OBJECT IDENTIFIER value of

{joint-iso-ccitt 100 3}

which is the same as

{2 100 3}

has a first subidentifier of 180 and a second subidentifier of 3. The resulting encoding is

OBJECT IDENTIFIER	Length	Contents
06 ₁₆	03 ₁₆	813403 ₁₆

8.20 Encoding for values of the restricted character string types

8.20.1 The data value consists of a string of characters from the character set specified in the ASN.1 type definition.

8.20.2 Each data value shall be encoded independently of other data values of the same type.

8.20.3 Each character string type shall be encoded as if it had been declared

[UNIVERSAL x] IMPLICIT OCTET STRING

where x is the number of the universal class tag assigned to the character string type in ITU-T Rec. X.680 | ISO/IEC 8824-1. The value of the octet string is specified in 8.20.4 and 8.20.5.

8.20.4 Where a character string type is specified in ITU-T Rec. X.680 | ISO/IEC 8824-1 by direct reference to an enumerating table (NumericString and PrintableString), the value of the octet string shall be that specified in 8.20.5 for a VisibleString type with the same character string value.

8.20.5 For restricted character strings apart from UniversalString and BMPString the octet string shall contain the octets specified in ISO 2022 for encodings in an 8-bit environment, using the escape sequence and character codings registered in accordance with ISO 2375.

8.20.5.1 An escape sequence shall not be used unless it is one of those specified by one of the registration numbers used to define the character string type in ITU-T Rec. X.680 | ISO/IEC 8824-1.

8.20.5.2 At the start of each string, certain registration numbers shall be assumed to be designated as G0 and/or C0 and/or C1, and invoked (using the terminology of ISO 2022). These are specified for each type in Table 3, together with the assumed escape sequence they imply.

8.20.5.3 Certain character string types shall not contain explicit escape sequences in their encodings; in all other cases, any escape sequence allowed by 8.20.5.1 can appear at any time, including at the start of the encoding. Table 3 lists the types for which explicit escape sequences are allowed.

8.20.5.4 Announcers shall not be used unless explicitly permitted by the user of ASN.1.

NOTE – The choice of ASN.1 type provides a limited form of announcer functionality. Specific application protocols may choose to carry announcers in other protocol elements, or to specify in detail the manner of use of announcers.

Table 3 – Use of escape sequences

Type	Assumed G0 (Registration number)	Assumed C0 & C1 (Registration number)	Assumed escape sequence(s) and locking shift (where applicable)	Explicit escape sequences allowed?
NumericString	6	None	ESC 2/8 4/2 LS0	No
PrintableString	6	None	ESC 2/8 4/2 LS0	No
TeletexString (T61String)	102	106 (C0) 107 (C1)	ESC 2/8 7/5 LS0 ESC 2/1 4/5 ESC 2/2 4/8	Yes
VideotexString	2	1 (C0) 73 (C1)	ESC 2/8 7/5 LS0 ESC 2/1 4/0 ESC 2/2 4/1	Yes
VisibleString (ISO646String)	6	None	ESC 2/8 4/2 LS0	No
IA5String	6	1 (C0)	ESC 2/8 4/2 LS0 ESC 2/1 4/0	No
GraphicString	6	None	ESC 2/8 4/2 LS0	Yes
GeneralString	6	1 (C0)	ESC 2/8 4/2 LS0 ESC 2/1 4/0	Yes

NOTE – Many of the commonly used characters (for example, A-Z) appear in a number of character repertoires with individual registration numbers and escape sequences. Where ASN.1 types allow escape sequences, a number of encodings may be possible for a particular character string (see also 7.3).

Example – With the ASN.1 type definition

Name ::= VisibleString

a value

"Jones"

can be encoded (primitive form) as

VisibleString	Length	Contents
1A ₁₆	05 ₁₆	4A6F6E6573 ₁₆

or (constructor form, definite length) as

VisibleString	Length	Contents												
3A ₁₆	09 ₁₆	<table> <thead> <tr> <th>OctetString</th> <th>Length</th> <th>Contents</th> </tr> </thead> <tbody> <tr> <td>04₁₆</td> <td>03₁₆</td> <td>4A6F6E₁₆</td> </tr> <tr> <th>OctetString</th> <th>Length</th> <th>Contents</th> </tr> <tr> <td>04₁₆</td> <td>02₁₆</td> <td>6573₁₆</td> </tr> </tbody> </table>	OctetString	Length	Contents	04 ₁₆	03 ₁₆	4A6F6E ₁₆	OctetString	Length	Contents	04 ₁₆	02 ₁₆	6573 ₁₆
OctetString	Length	Contents												
04 ₁₆	03 ₁₆	4A6F6E ₁₆												
OctetString	Length	Contents												
04 ₁₆	02 ₁₆	6573 ₁₆												

or (constructor form, indefinite length) as

VisibleString	Length	Contents																		
3A ₁₆	80 ₁₆	<table> <thead> <tr> <th>OctetString</th> <th>Length</th> <th>Contents</th> </tr> </thead> <tbody> <tr> <td>04₁₆</td> <td>03₁₆</td> <td>4A6F6E₁₆</td> </tr> <tr> <th>OctetString</th> <th>Length</th> <th>Contents</th> </tr> <tr> <td>04₁₆</td> <td>02₁₆</td> <td>6573₁₆</td> </tr> <tr> <th>EOC</th> <th>Length</th> <th></th> </tr> <tr> <td>00₁₆</td> <td>00₁₆</td> <td></td> </tr> </tbody> </table>	OctetString	Length	Contents	04 ₁₆	03 ₁₆	4A6F6E ₁₆	OctetString	Length	Contents	04 ₁₆	02 ₁₆	6573 ₁₆	EOC	Length		00 ₁₆	00 ₁₆	
OctetString	Length	Contents																		
04 ₁₆	03 ₁₆	4A6F6E ₁₆																		
OctetString	Length	Contents																		
04 ₁₆	02 ₁₆	6573 ₁₆																		
EOC	Length																			
00 ₁₆	00 ₁₆																			

8.20.6 The above example illustrates three of the (many) possible forms available as a sender's option. Receivers are required to handle all permitted forms (see 7.3).

8.20.7 For the "UniversalString" type, the octet string shall contain the octets specified in ISO/IEC 10646-1, using the 4-octet canonical form (see 14.2 of ISO/IEC 10646-1). Control functions and signatures shall not be used.

8.20.8 For the "BMPString" type, the octet string shall contain the octets specified in ISO/IEC 10646-1, using the 2-octet BMP form (see 14.1 of ISO/IEC 10646-1). Control functions and signatures shall not be used.

8.21 Encoding for values of the unrestricted character string type

8.21.1 There are two sub-rules used for encoding the unrestricted character type, called CH-A (the index-setting rule) and CH-B (the index-using rule). For any given value of "identification" in the abstract value, the first occurrence (within the encoding of an entire pdv) of a value of the unrestricted character string type with this value of "identification" shall be encoded using sub-rule CH-A.

8.21.2 Subject to certain restrictions listed below, subsequent values with the same value of "identification" shall be encoded using sub-rule CH-B with the "index" set to the same value as the "index" in the corresponding CH-A encoding.

8.21.3 The conditions for use of sub-rule CH-B are listed below. If any condition is not satisfied, then sub-rule CH-A shall be used instead. The conditions for use of sub-rule CH-B are:

- a) the "index" value is in the range 0 to 255;
- b) the length of the encoding is not greater than the maximum length that can be identified by a long-form length encoding.

NOTE – The last condition is not likely to prove a restriction in practice.

8.21.4 For the first occurrence of a CH-A encoding the "index" shall have the value zero, and for each subsequent occurrence it shall be incremented by one.

8.21.5 TUTORIAL – Thus, for any given "identification" value there is a (relatively inefficient) CH-A encoding with a unique index value and carries the full "identification" value, followed by arbitrarily many (efficient) CH-B encodings linked to the CH-A encoding by the index value. Because the CH-B encoding uses a single octet for the index, and a count in octets for the encoding, it cannot be used if the index value exceeds 255 (256 different "identifications" are in use). In this case the CH-A encoding is used for all such occurrences.

8.21.6 The CH-A encoding shall be the BER encoding of the following sequence type after the application of the AUTOMATIC TAGS as specified in ITU-T Rec. X.680 | ISO/IEC 8824-1, subclauses 22.6 and 26.3:

```
[UNIVERSAL 29] IMPLICIT SEQUENCE {
  index          INTEGER,
  identification CHOICE {
    syntaxes     SEQUENCE {
      abstract   OBJECT IDENTIFIER,
      transfer   OBJECT IDENTIFIER },
    syntax       OBJECT IDENTIFIER,
    presentation-context-id INTEGER,
    context-negotiation SEQUENCE {
      presentation-context-id INTEGER,
      transfer-syntax       OBJECT IDENTIFIER },
    transfer-syntax OBJECT IDENTIFIER,
    fixed           NULL },
  string-value    OCTET STRING }
```

8.21.7 The value of "string-value" shall be the encoding of the abstract character string value using the identified character transfer syntax, the value of "index" shall be as determined above, and the value of all other fields shall be the same as the values appearing in the abstract value.

NOTES

1 The component "index" is not defined in the abstract syntax because this is meant to be supplied strictly at the encoding rule level (in the same sense that end-of-contents octets are meant to be applied at the encoding rule level).

2 Both alternatives of "string-value" in the abstract syntax are encoded identically – as a bitstring – in the transfer syntax.

8.21.8 The CH-B encoding shall be the BER encoding of the following ASN.1 type:

```
[UNIVERSAL 29] IMPLICIT OCTET STRING
```

where

- a) the encoding is primitive;
- b) the length octets shall be the short or long definite form as a sender's option;

- c) the contents octets shall encode the "index" in the first octet of the octet string value as an integer that ranges in value from 0 through 255, followed by the encoding of the "string-value".

NOTE – A receiver can distinguish the CH-A encoding from the CH-B encoding by the setting of the primitive/constructor bit.

9 Canonical encoding rules

The encoding of a data values employed by the canonical encoding rules is the basic encoding described in clause 8, together with the following restrictions and those also listed in clause.

9.1 Length forms

If the encoding is constructed, it shall employ the indefinite length form. If the encoding is primitive, it shall include the fewest length octets necessary. (Contrast with 8.1.3.2 b.)

9.2 String encoding forms

Bitstring, octetstring, and restricted character string values shall be encoded with a primitive encoding if they would require no more than 1000 contents octets, and as a constructed encoding otherwise. The string fragments contained in the constructed encoding shall be encoded with a primitive encoding. The encoding of each fragment, except possibly the last, shall have 1000 contents octets. (Contrast with 8.20.6.)

9.3 Set components

The encodings of the component values of a set value shall appear in an order determined by their tags as specified in 6.4 of ITU-T Rec. X.680 | ISO/IEC 8824-1. Additionally, for the purposes of determining the order in which components are encoded when one or more component is an untagged choice type, each untagged choice type is ordered as though it has a tag equal to that of the smallest tag in that choice type or any untagged choice types nested within.

Example – In the following which assumes a tagging environment of IMPLICIT TAGS

```
A ::= SET
{
  a    [3] INTEGER,
  b    [1] CHOICE
  {
    c    [2] INTEGER,
    d    [4] INTEGER
  },
  e    CHOICE
  {
    f    CHOICE
    {
      g    [5] INTEGER,
      h    [6] INTEGER
    },
    i    CHOICE
    {
      j    [0] INTEGER
    }
  }
}
```

the order in which the components of the set are encoded will always be e, b, a, since the tag [0] sorts lowest, then [1], then [3].

10 Distinguished encoding rules

The encoding of a data values employed by the distinguished encoding rules is the basic encoding described in clause 8, together with the following restrictions and those also listed in clause.

NOTE – Recommendation X.509 | ISO/IEC 9594-8 forbids the use of base 10 abstract values in the Directories applications

10.1 Length forms

The definite form of length encoding shall be used, encoded in the minimum number of octets. (Contrast with 8.1.3.2 b.)

10.2 String encoding forms

For bitstring, octetstring and restricted character string types, the constructed form of encoding shall not be used. (Contrast with 8.20.6.)

10.3 Set components

The encodings of the component values of a set value shall appear in an order determined by their tags as specified in 6.4 of ITU-T Rec. X.680 | ISO/IEC 8824-1.

NOTE – Where a component of the set is an untagged choice type, the location of that component in the ordering will depend on the tag of the choice component being encoded.

11 Restrictions on BER employed by both CER and DER

References in clause 8 and its subclauses to "shall be the BER encoding" shall be interpreted as "shall be the CER or DER encoding, as appropriate". (See 8.16.1, 8.17.6, 8.17.8, 8.18.1, 8.21.8 and 8.21.6.)

11.1 Boolean values

If the encoding represents the boolean value TRUE, its single contents octet shall have all eight bits set to one. (Contrast with 8.2.2.)

11.2 Unused bits

11.2.1 Each unused bit in the final octet of the encoding of a bit string value shall be set to zero.

11.2.2 Where ITU-T Rec. X.680 | ISO/IEC 8824-1, subclause 19.7 applies, the bitstring shall have all trailing 0 bits removed before it is encoded.

NOTES

1 In the case where a size constraint has been applied, the abstract value delivered by a decoder to the application will be one of those satisfying the size constraint and differing from the transmitted value only in the number of trailing 0 bits.

2 If a bitstring value has no 1 bits then an encoder shall encode the value with a length of 1 and an initial octet set to 0.

11.3 Real values

11.3.1 If the encoding represents a real value whose base B is 2, then binary encoding employing base 2 shall be used. Before encoding, the mantissa M and exponent E are chosen so that M is either 0 or is odd.

NOTE – This is necessary because the same real value can be regarded as both {M, 2, E} and {M', 2, E'} with $M \neq M'$ if, for some non-zero integer n:

$$M' = M \times 2^{-n}$$

$$E' = E + n$$

In encoding the value, the binary scaling factor F shall be zero, and M and E shall each be represented in the fewest octets necessary.

11.3.2 If the encoding represents a real value whose base B is 10, then decimal encoding shall be used. In forming the encoding, the following applies:

11.3.2.1 The ISO 6093 NR3 form shall be used (see 8.5.6).

11.3.2.2 SPACE shall not be used within the encoding.

11.3.2.3 If the real value is negative, then it shall begin with a MINUS SIGN (–), otherwise, it shall begin with a digit.

11.3.2.4 Neither the first nor the last digit of the mantissa may be a 0.

11.3.2.5 The last digit in the mantissa shall be immediately followed by FULL STOP (.), followed by the exponent-mark "E".

11.3.2.6 If the exponent has the value 0, it shall be written "+0", otherwise the exponent's first digit shall not be zero, and PLUS SIGN shall not be used.

11.4 GeneralString values

The encoding of values of the GeneralString type (and its subtypes) shall generate escape sequences to designate and invoke a new register entry only when the register entry for the character is different from that currently designated as G0, C0, or C1. All designations and invocations shall be into the G0 set or the C0 set.

NOTE – It is assumed that each character in a character string value is associated with a particular entry in the International Register of Coded Character Sets.

11.5 Set and sequence components with default value

The encoding of a set value or sequence value shall not include an encoding for any component value which is equal to its default value.

11.6 Set-of components

The encodings of the component values of a set-of value shall appear in ascending order, the encodings being compared as octet strings.

11.7 GeneralizedTime

11.7.1 The encoding shall terminate with a "Z", as described in the CCITT X.208-1 | ISO/IEC 8824-1 clause on GeneralizedTime.

11.7.2 The fractional-seconds elements, if present, shall omit all trailing 0's; if the elements correspond to 0, they shall be wholly omitted, and the decimal point element also shall be omitted.

Example – A seconds element of "26.000" shall be represented as "26"; a seconds element of "26.5200" shall be represented as "26.52".

11.7.3 The decimal point element, if present, shall be the point option ".".

11.7.4 Midnight (GMT) shall be represented in the form:

"YYYYMMDD000000Z"

where "YYYYMMDD" represents the day following the midnight in question.

11.7.5 Examples of valid representations

"19920521000000Z"

"19920622123421Z"

"19920722132100.3Z"

11.7.6 Examples of invalid representations

"19920520240000Z" (midnight represented incorrectly)

"19920622123421.0Z" (spurious trailing 0's)

"19920722132100.30Z" (spurious trailing 0's)

12 Use of BER, CER and DER in transfer syntax definition

12.1 The encoding rules specified in this Recommendation | International Standard can be referenced and applied whenever there is a need to specify an unambiguous, undivided and self-delimiting octet string representation for all of the values of a single ASN.1 type.

NOTE – All such octet strings are unambiguous within the scope of the single ASN.1 type. They would not necessarily be unambiguous if mixed with encodings of a different ASN.1 type.

12.2 The following object identifier and object descriptor values are assigned to identify and describe the basic encoding rules specified in the Recommendation | International Standard:

{joint-iso-ccitt asn1 (1) basic-encoding (1)}

and

"Basic Encoding of a single ASN.1 type"

12.3 The following object identifier and object descriptor values are assigned to identify and describe the canonical encoding rules specified in the Recommendation | International Standard:

{joint-iso-ccitt asn1(1) ber-derived(2) canonical-encoding(0)}

and

"Canonical encoding of a single ASN.1 type"

12.4 The following object identifier and object descriptor values are assigned to identify and describe the distinguished encoding rules specified in the Recommendation | International Standard:

{joint-iso-ccitt asn1(1) ber-derived(2) distinguished-encoding(1)}

and

"Distinguished encoding of a single ASN.1 type"

12.5 Where an unambiguous specification defines an abstract syntax as a set of presentation data values, each of which is a value of some specifically named ASN.1 type, usually (but not necessarily) a choice type, then one of the object identifier values specified in 12.1, 12.3 or 12.4 may be used with the abstract syntax name to identify the basic encoding rules, canonical encoding rules or distinguished encoding rules, respectively, to the specifically named ASN.1 type used in defining the abstract syntax.

12.6 The names specified in 12.1, 12.3 and 12.4 shall not be used with an abstract syntax name to identify a transfer syntax unless the conditions of 12.5 for the definition of the abstract syntax (see ITU-T Rec. X.680 | ISO/IEC 8824-1, subclause D.3) are met.

Annex A

Example of encodings

(This annex does not form an integral part of this Recommendation | International Standard)

This annex illustrates the basic encoding rules specified in this Recommendation | International Standard by showing the representation in octets of a (hypothetical) personnel record which is defined using ASN.1

A.1 ASN.1 description of the record structure

The structure of the hypothetical personnel record is formally described below using ASN.1 specified in ITU-T Rec. X.680 | ISO/IEC 8824-1 for defining types.

```

PersonnelRecord ::= [APPLICATION 0] IMPLICIT SET {
    name          Name,
    title         [0] VisibleString,
    number        EmployeeNumber,
    dateOfHire    [1] Date,
    nameOfSpouse  [2] Name,
    children      [3] IMPLICIT
        SEQUENCE OF ChildInformation DEFAULT {} }

ChildInformation ::= SET
    { name          Name,
      dateOfBirth  [0] Date}

Name ::= [APPLICATION 1] IMPLICIT SEQUENCE
    {givenName     VisibleString,
     initial       VisibleString,
     familyName    VisibleString}

EmployeeNumber ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT VisibleString -- YYYYMMDD

```

A.2 ASN.1 description of a record value

The value of John Smith's personnel record is formally described below using ASN.1.

```

{ name {givenName "John",initial "P",familyName "Smith"},
  title "Director",
  number 51,
  dateOfHire "19710917",
  nameOfSpouse {givenName "Mary",initial "T",familyName "Smith"},
  children
    {{{givenName "Ralph",initial "T",familyName "Smith"},
      dateOfBirth "19571111"},
    {{givenName "Susan",initial "B",familyName "Jones"},
      dateOfBirth "19590717"}}}

```

A.3 Representation of this record value

The representation in octets of the record value given above (after applying the basic encoding rules defined in this Recommendation | International Standard) is shown below. The values of identifiers, lengths, and the contents of integers are shown in hexadecimal, two hexadecimal digits per octet. The values of the contents of character strings are shown as text, one character per octet.

Personnel

Record	Length	Contents
60	8185	

Name	Length	Contents
61	10	

VisibleString	Length	Contents
1A	04	"John"
VisibleString	Length	Contents
1A	01	"P"

Superseded by a more recent version ISO/IEC 8825-1 : 1995 (E)

		VisibleString	Length	Contents				
		1A	05	"Smith"				
Title	Length	Contents						
A0	0A							
		VisibleString	Length	Contents				
		1A	08	"Director"				
Employee								
Number	Length	Contents						
42	01	33						
Date of								
Hire	Length	Contents						
A1	0A							
		Date	Length	Contents				
		43	08	"19710917"				
Name of								
Spouse	Length	Contents						
A2	12							
		Name	Length	Contents				
		61	10					
		VisibleString	Length	Contents				
		1A	04	"Mary"				
		VisibleString	Length	Contents				
		1A	01	"T"				
		VisibleString	Length	Contents				
		1A	05	"Smith"				
[3]	Length	Contents						
A3	42							
		Set	Length	Contents				
		31	1F					
		Name	Length	Contents				
		61	11					
		VisibleString	Length	Contents				
		1A	05	"Ralph"				
		VisibleString	Length	Contents				
		1A	01	"T"				
		VisibleString	Length	Contents				
		1A	05	"Smith"				
		Date of						
		Birth	Length	Contents				
		A0	0A					
		Date	Length	Contents				
		43	08	"19571111"				
		Set	Length	Contents				
		31	1F					
		Name	Length	Contents				
		61	11					
		VisibleString	Length	Contents				
		1A	05	"Susan"				
		VisibleString	Length	Contents				
		1A	01	"B"				
		VisibleString	Length	Contents				
		1	05	"Jones"				
		Date of						
		Birth	Length	Contents				
		A0	0A					
		Date	Length	Contents				
		43	08	"19590717"				

Annex B

Assignment of object identifier values

(This annex does not form an integral part of this Recommendation | International Standard)

The following values are assigned in this Recommendation | International Standard:

Clause Object Identifier Value

12.2 {joint-iso-ccitt asn1 (1) basic-encoding (1)}

Object Descriptor Value

"Basic Encoding of a single ASN.1 type"

Clause Object Identifier Value

12.3 {joint-iso-ccitt asn1(1) ber-derived(2) canonical-encoding(0)}

Object Descriptor Value

"Canonical encoding of a single ASN.1 type"

Clause Object Identifier Value

12.4 {joint-iso-ccitt asn1(1) ber-derived(2) distinguished-encoding(1)}

Object Descriptor Value

"Distinguished encoding of a single ASN.1 type"

Annex C

Illustration of real value encoding

(This annex does not form an integral part of this Recommendation | International Standard)

C.1 A sender will normally examine his own hardware floating point representation to determine the (value-independent) algorithms to be used to transfer values between this floating-point representation and the length and contents octets of the encoding of an ASN.1 real value. This annex illustrates the steps which could be taken in such a process by using the (artificial) hardware floating point representation of the mantissa shown in Figure C-1.

It is assumed that the exponent can easily be obtained from the floating point hardware as an integer value E.

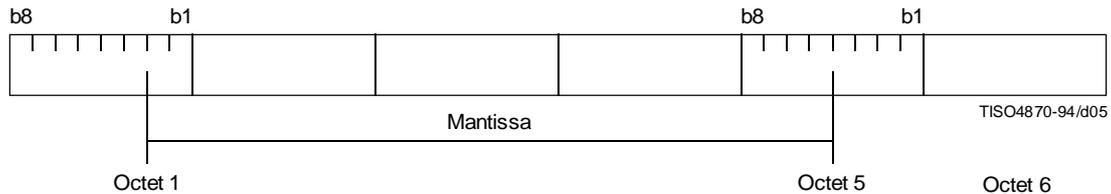


Figure C-1 – Floating point representation

C.2 The contents octets which need to be generated for sending a non-zero value using binary encoding (as specified in the body of this Recommendation | International Standard) are:

$$1\ S\ bb\ ff\ ee\ \text{Octets for E}\ \text{Octets for N}$$

where S (the mantissa sign) is dependent on the value to be converted, bb is a fixed value (say 10) to represent the base (in this case let us assume base 16), ff is the fixed F value calculated as described in C.3, and ee is a fixed length of exponent value calculated as described in C.4. (This annex does not treat the case where E needs to exceed three octets.)

C.3 The algorithm will transmit octets 1 to 5 of the hardware representation as the value of N, after forcing bits 8 to 3 of octet 1 and bits 4 to 1 of octet 5 to zero. The implied decimal point is assumed to be positioned between bits 2 and 1 of octet 1 in the hardware representation which delivers the value of E. Its implied position can be shifted to the nearest point after the end of octet 5 by reducing the value of E before transmission. In our example system we can shift by four bits for every exponent decrement (because we are assuming base 16), so a decrement of 9 will position the implied point between bits 6 and 5 of octet 6. Thus the value of M is N multiplied by 2^3 to position the point correctly in M. (The implied position in N, the octets transferred, is after bit 1 of octet 5.) Thus we have the crucial parameters:

$$F = 3 \quad (\text{so } ff \text{ is } 11)$$

$$\text{exponent decrement} = 9$$

C.4 The length needed for the exponent is now calculated by working out the maximum number of octets needed to represent the values

$$E_{\min} - \text{excess} - \text{exponent decrement}$$

$$E_{\max} - \text{excess} - \text{exponent decrement}$$

where E_{\min} and E_{\max} are minimum and maximum integer values of the exponent representation, excess is any value which needs subtracting to produce the true exponent value, and the exponent decrement is as calculated in C.3. Let us assume this gives a length of 3 octets. Then ee is 10. Let us also assume excess is zero.

C.5 The transmission algorithm is now:

- a) Transmit the basic encoding rules identifier octets field with a tag for ASN.1 type real.
- b) Test for zero, and if so, transmit an ASN.1 basic encoding rules length field with value of zero (no contents octets), and end the algorithm.

- c) Test and remember the mantissa sign, and negate the mantissa if negative.
- d) Transmit an ASN.1 basic encoding rules length field with value of 9, then:
 - 11101110, if negative; or
 - 10101110, if positive.
- e) Produce and transmit the 3 octet exponent with value
$$E - 9$$
- f) Zero bits 8 to 3 of octet 1 and bits 4 to 1 of octet 5, then transmit the 5 octet mantissa.

C.6 The receiving algorithm has to be prepared to handle any ASN.1 basic encoding, but here the floating point unit can be directly used. We proceed as follows:

- a) Check octet 1 of the contents; if it is 1x101110 we have a transmission compatible with ours, and can simply reverse the sending algorithm.
- b) Otherwise, for character encoding, invoke standard character decimal to floating point conversion software, and deal with a "SpecialRealValue" according to the application semantics (perhaps setting the largest and smallest number the hardware floating point can handle).
- c) For a binary transmission, put N into the floating point unit, losing octets at the least significant end if necessary, multiply by 2^F , and by B^E , then negate if necessary. Implementors may find optimization possible in special cases, but may find (apart from the optimization relating to transmissions from a compatible machine) that testing for them loses more than they gain.

C.7 The above algorithms are illustrative only. Implementors will, of course, determine their own best strategies.

Annex D

Use of the DER and CER in data origin authentication

(This annex does not form an integral part of this Recommendation | International Standard)

D.1 The problem to be solved

D.1.1 The distinguished encoding rules and canonical encoding rules have been provided to assist in the provision of integrity security mechanisms using authenticators for material to be transferred.

NOTE – Throughout the rest of this annex only DER is mentioned for purposes of simplicity. However, the text applies just as well to CER.

D.1.2 The concept of an authenticator is well understood, and involves taking the bit pattern to be transferred, applying some form of hashing function to it to reduce it to a few octets, encrypting those octets to authenticate the authenticator, then transmitting the authenticator with the original material (the original material being sent in clear). On receipt, the authenticator is recalculated from the received clear text and compared with the received authenticator. If they are equal, the text was not tampered with, otherwise it was.

D.1.3 This simple concept becomes more difficult when the ISO model, and particularly the presentation layer, is in use.

D.1.4 Two problems arise, one of which is a question of modelling and so-called layer independence, and the second of which relates to the use of application layer relays, such as are used in Recommendation X.400.

D.1.5 On the modelling issue, the hash function and the encryption algorithm are part of the application's operation, but the application has no knowledge or control of the actual encoding which the presentation layer will use. Similarly on receipt, decoding, and hence destruction of the bit string on receipt is a presentation layer matter. There are four solutions that have been proposed to overcome this problem:

- a) Rule out of order the use of the actual octets produced by the presentation layer for use in the authenticator; (the current philosophy being adopted by presentation and ULA experts).
- b) Put the hashing and authenticator mechanisms into the presentation layer itself (this solution was rejected as part of the broad question of putting support for encryption into ASN.1. At the time of rejection, the reason for the rejection was that work on security was still immature, and that one did not want to prejudice the eventual result).
- c) Model a complex interaction with the presentation layer where, on transmission, a value is presented for encoding, the encoding is produced and returned to the application layer which calculates the authenticator, then the whole is transmitted. On receipt, as well as producing the abstract value, the received encodings are passed to the application layer for authenticator checking (this model was rejected by the ULA group).
- d) Do the entire encoding in the application layer, and make no use of presentation services for transfer syntax negotiation (this is really a rejection of the OSI reference model, and would not be acceptable as a wide-spread solution).

D.1.6 It might be argued that failure to agree on a model to describe an apparently simple and workable process (produce the encoding, then the authenticator, and transmit both, check against the authenticator on receipt) is not something which should be accepted as a long-term position. Such a remark would have strong validity if it were not for the second problem of application relays, and if there were no other workable solution. (This annex is outlining an alternative solution which is used in CCITT Rec. X.509 | ISO/IEC 9594-8 and is considered to be free from modelling and relay-system problems and workable.)

D.1.7 The second problem is that, if an application relay is in place, the transfer syntax used for the second transmission may be different from that agreed for the first (for example, use of PER on one of them, and BER on the other). This would defeat the authenticator, unless the authenticator was opened up and recalculated at the relay, which would imply security exchanges with the relay, whereas what is required is end-to-end security.

NOTE – There have been suggestions that one might want to flag a presentation context as "do not decode/re-encode at application relays", but this also provides modelling and other problems.

D.1.8 Thus, we are led to try to work with a model in which the presentation layer (together with any intervening application relays) provides for the transfer of the abstract syntax and semantics of the information, but makes no guarantees that the actual bit-pattern encoding (the transfer syntax) will be retained end-to-end.

D.1.9 The challenge is therefore to provide an authenticator mechanism that can operate on the abstract data type, rather than on the transmitted bit string.

D.1.10 The Directories group were the first to attempt to produce a solution to this problem, and it is their model that is described below.

D.2 The approach to a solution

D.2.1 The following text describes first a conceptual model of what is being done, followed by an implementation optimization that eliminates the double encoding/decoding implied in the conceptual model.

D.2.2 The conceptual model works as follows:

- a) The sender, in the application layer, converts the abstract value into a bitstring using the DER, and produces the authenticator from that bitstring, which is added to the abstract value, and both values are transmitted using normal presentation layer mechanisms, and any transfer syntax. Conceptually, the sender is encoding twice – once for the authenticator (using the DER) in the application layer, and once for the actual transfer (using the negotiated transfer syntax) in the presentation layer.

NOTE – The important property of the bitstring produced by the DER is that it is in one-to-one correspondence with the abstract value. Thus, end-to-end transfer without loss of information at the abstract syntax level is equivalent to end-to-end transfer of the bitstring on which the authenticator is based.

- b) The receiver will decode the received bitstring in the presentation layer, using the negotiated transfer syntax (which may differ from that used by the sender if an application relay is in place), and will pass the abstract value to the application. In the application layer, the abstract value is re-encoded using the DER to produce the bitstring to be authenticated.

D.2.3 Thus, conceptually, we encode twice at the sending end, and decode once and then encode at the receiving end. Implementors may choose to actually do this if the code supporting presentation layer operation is from a supplier different from that producing the code to support the application. How significant an overhead this would be is not clear at this stage. Where an integrated implementation is used, however, there is the option of the optimization described below. It should also be noted that the DER are no harder to apply than the BER except in relation to use of set-of. If a large set-of is to be processed, the implementation may need to invoke a disk-based sorting routine. Application designers should be aware of this, and try to use sequence-of instead of set-of when use of the DER is envisaged.

D.3 The implementation optimization

D.3.1 The OSI model and protocol standards specify required behavior, but they do not, in any way, seek to constrain the architecture and structure of actual implementation code. Thus an implementor can produce the desired effect in whatever way she chooses.

D.3.2 At the sending end, the bitstring which is produced (conceptually in the application layer) can be kept, and used to provide the encoding that is conceptually performed in the presentation layer. This is suitable for sending if the negotiated transfer syntax is either the ASN.1 BER or DER. If it is neither of these, then double encoding is necessary.

D.3.3 Similarly, at the receiving end, the received bitstring can be retained (for any transfer syntax), and the implementation can use this to check the authenticator. If it matches, end of problem. If it does not match, then it may be a transfer syntax problem, and recoding from the abstract value is necessary to determine whether there was tampering or not.

D.3.4 In order to maximize the chances of not having to do double encoding/decoding, systems using this mechanism are advised to try to negotiate a transfer syntax of DER (using the appropriate object identifier) as their first preference, falling back onto BER or some other encoding rules.