INTERNATIONAL TELECOMMUNICATION UNION

ITU-T X.209

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

OPEN SYSTEMS INTERCONNECTION
MODEL AND NOTATION

SPECIFICATION OF BASIC ENCODING RULES FOR ABSTRACT SYNTAX NOTATION ONE (ASN.1)

ITU-T Recommendation X.209

(Extract from the *Blue Book*)

NOTES

1 ITU-T Recommendation X.209 was published in Fascicle VIII.4 of the *Blue Book*. This file is an extract from the *Blue Book*. While the presentation and layout of the text might be slightly different from the *Blue Book* version, the contents of the file are identical to the *Blue Book* version and copyright conditions remain unchanged (see below).

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

Recommendation X.209

SPECIFICATION OF BASIC ENCODING RULES FOR ABSTRACT SYNTAX NOTATION ONE (ASN.1)[1)]

(Melbourne, 1988)

The CCITT,

considering

(a) the variety and complexity of information objects conveyed within the application layer;

(b) the need for a high-level notation for specifying such information objects;

(c) the value of isolating and standardizing the rules for encoding such information objects.

unanimously recommends

that the rules for encoding information objects are defined in this Recommendation.

CONTENTS

———————

1) Recommendation X.209 and ISO 8825 [Information processing systems - Open systems interconnection - Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1) as extended by Addendum 1 to ISO 8825, were developed in close cooperation and are technically aligned.

0 Introduction

Recommendation X.208 (Specification of Abstract Syntax Notation One) specifies a notation for the definition of abstract syntaxes, enabling application layer specifications to define the types of information they need to transfer using the presentation service. It also specifies a notation for the specification of value of a defined type.

This Recommendation defines a set of encoding rules that may be applied to values of types defined using the notation specified in Recommendation X.208. 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 notation of Recommendation X.208. This Recommendation defines one set of encoding rules, called **basic encoding rules**.

This Recommendation is technically and editorially aligned with ISO 8825 plus Addendum I to ISO 8825.

Appendix I gives examples of the application of the encoding rules. It is not part of this Recommendation.

Appendix II summarises the assignment of object identifier values made in this Recommendation and is not part of this Recommendation.

Appendix III is not part of this Recommendation, and gives examples of applying the rules for encoding reals.

1 Scope and field of application

This Recommendation 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 Recommendation X.208. These basic encoding rules are also to be applied for decoding such a transfer syntax in order to identify the data values being transferred.

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

2 References

[1] Recommendation X.200, Reference Model of Open Systems Interconnection for CCITT Applications (see also ISO 7498).

[2] Recommendation X.208, Specification of Abstract Syntax Notation One (ASN.1) (see also ISO 8824).

[3] Recommendation X.226, Presentation Protocol Specification for Open Systems Interconnection for CCITT Applications (see also ISO 8823).

[4] ISO 2022, Information processing - ISO 7-bit and 8-bit coded character sets - Code extension techniques.

[5] ISO 2375, Data processing - Procedure for registration of escape sequences.

[6] ISO 6093, Information processing - Representation of numerical values in character strings for information interchange.

3 Definitions

The definitions of Recommendation X.208 are used in this Recommendation.

3.1 dynamic conformance

A statement of the requirement for an implementation to adhere to the behaviour prescribed by this Recommendation 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.

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.

Note - Some CCITT Recommendations use the term "data element" for this sequence of octets, but the term is not used in this Recommendation, as ISO International Standard use it to mean "data value".

3.5 identifier octets

Part of a data value encoding which is used to identify the type of the 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 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.

3.8 contents octets

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

3.9 **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 other data values.

3.11 **sender**

An implementation encoding a data value for transfer.

3.12 **receiver**

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

4 Abbreviations and notation

4.1 *Abbreviations*

ASN.1 Abstract Syntax Notation One

4.2 *Notation*

4.2.1 This Recommendation references the notation defined by Recommendation X.208.

4.2.2 This Recommendation 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.

4.2.3 For the purposes of this Recommendation, 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".

5 Conformance

5.1 Dynamic conformance is specified by § 6 to § 24 inclusive.

5.2 Static conformance is specified by those documents which specify the application of these basic encoding rules.

5.3 Alternative encodings are permitted. by this Recommendation as a sender's option. Conforming receivers shall support all alternatives.

Note - Examples of such alternative encodings appear in § 6.3.2 b) and Table 2/X.209.

6 General rules for encoding

6.1 *Structure of an encoding*

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

a) identifier octets (see § 6.2);

b) length octets (see § 6.3);

c) contents octets (see § 6.4);

d) end-of-contents octets (see § 6.5).

6.1.2 The end-of-contents octets shall not be present unless the value of the length octets requires them to be present (see § 6.3).

6.1.3 Figure 1/X.209 illustrates the structure of an encoding (primitive or constructed). Figure 2/X.209 illustrates an alternative constructed encoding.

6.2 *Identifier octets*

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

6.2.2 For tags with a number ranging from zero to 30 (inclusive), the identifier octets shall comprise a single

a) bits 8 and 7 shall be encoded to represent the class of the tag as specified in Table 1 /X.209;

b) bit 6 shall be a zero or a one according to the rules of § 6.2.5;

c) bit 5 to I shall encode the number of the tag as a binary integer with bit 5 as the most significant bit.

6.2.3 Figure 3/X.209 illustrates the form of an identifier octet for a type with a tag whose number is in the range zero to 30 (inclusive).

6.2.4 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.

TABLE 1/X.209

Encoding of clan of tag

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

6.2.4.1 The leading octet shall be encoded as follows:

a) bits 8 and 7 shall be encoded to represent the class of the tags as listed in Table 1 /X.209;

b) bit 6 shall be a zero or a one according to the rule of § 6.2.5;

c) bit 5 to I shall be encoded as 11111_2.

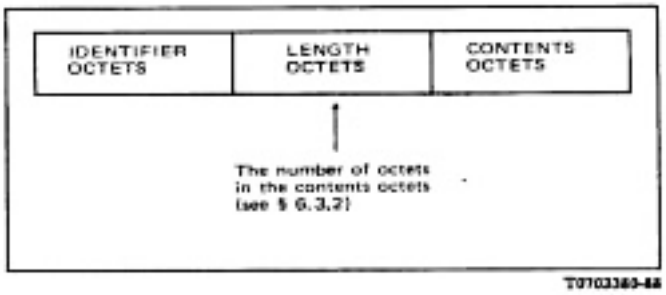

FIGURE 1/X.209

Structure of an encoding

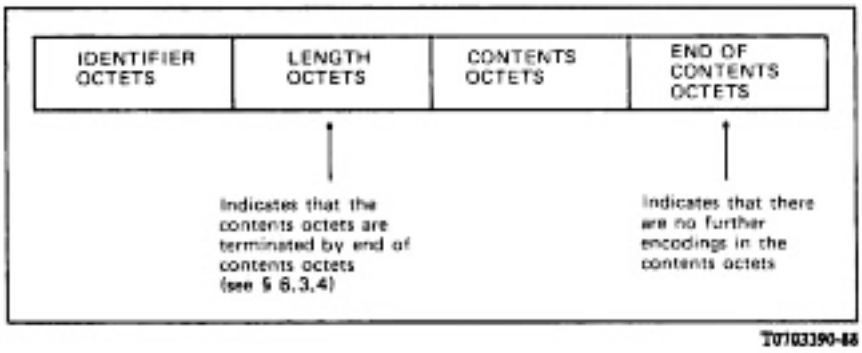

FIGURE 2/X.209

An alternative constructed encoding

6.2.4.2 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) bit 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.

6.2.4.3 Figure 4/X.209 illustrates the form of the identifier octets for a type with a tag whose number is greater than 30.

6.2.5 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.

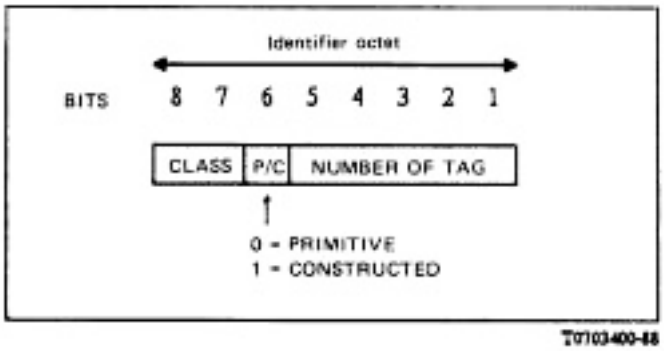

FIGURE 3/X209

Identifier octet (low tag number)

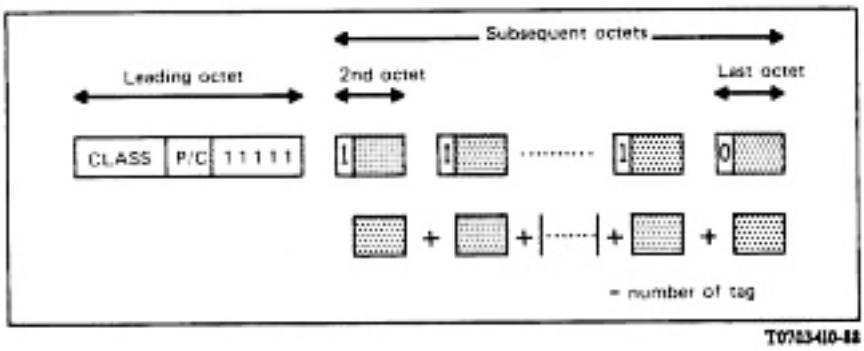

FIGURE 4/X.209

Identifier octets (high tag number)

6.2.6 Recommendation X.208 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.

6.2.7 Recommendation X.208 specifies that the tag of a type defined using "ANY" is indeterminate. The "ANY" 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).

6.3 *Length octets*

6.3.1 Two forms of length octets specified. These are

a) the definite form (see § 6.3.3); and

b) the indefinite form (see § 6.3.4).

6.3.2 A sender shall

a) use the definite form (§ 6.3.3) if the encoding is primitive;

b) use either the definite form (§ 6.3.3) or the indefinite form (§ 6.3.4), a sender's option, if the encoding is constructed and all immediately available;

c) use the indefinite form (§ 6.3.4) if the encoding is constructed and is not all immediately available.

6.3.3 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 (§ 6.3.3.1) or the long form (§ 6.3.3.2) 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.

6.3.3.1 In the short form, the length octets shall consist of a single octet in which bit 8 is zero and bit 7 to I 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 00100110_2

6.3.3.2 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 11111111_2 shall not be used

Note - 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: 10000001_2

11001001_2

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

6.3.4 For the indefinite form, the length octets indicate that the contents are terminated by end-of-contents octets (see § 6.5), and shall consist of a single octet.

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

6.3.4.2 If this form of length is used, then end-of-contents octets (see § 6.5) shall be present in the encoding following the contents octets.

6.4 *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.

6.5 *End-of-contents octets*

The end-of-contents octets shall be present if the length is encoded as specified in § 6.3.4, 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, whose form is primitive, whose number of the tag is zero, and whose contents is absent, thus

End-of-contents	Length	Contents
00_{16}	00_{16}	Absent

7 Encoding of a Boolean value

7.1 The encoding of a Boolean value shall be primitive. The contents octets shall consist of a single octet.

7.2 If the Boolean value is

FALSE

the octet shall be zero.

7.2.1 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 Encoding of an integer value

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

8.2 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 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.

9 Encoding of an enumerated value

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

10 Encoding of a real value

10.1 The encoding of a real value shall be primitive.

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

10.3 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 § 10.5, shall be used. If B' is 10, a character encoding, specified in § 10.6, shall he 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.

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

a) if bit 8 = 1, then the binary encoding specified in § 10.5 applies;

b) if bit 8 = 0 and bit 7 = 0, then the decimal encoding specified in § 10.6 applies;

c) if bit 8 = 0 and bit 7 = 1, then a 'SpecialRealValue" (see Recommendation X.208) is encoded as specified in § 10.7.

10.5 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 - This freedom to choose F is provided to enable easier generation of the transfer format by eliminating the need to align the implied decimal point of the mantissa with an octet boundary (see Appendix III). The existence of F does not noticeably complicate the task of receivers.

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

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

Bits 6 to 5	Base
00	base 2
01	base 8
10	base 16
11	Reserved for future versions of this Recommendation

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

10.5.4 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 exponents 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 plus 3)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.

10.5.5 The remaining contents octets encode the value of the integer N (see § 10.5) as an unsigned binary number.

Note 1 - This encoding does not specify a "normalized" representation, there being a number of possible representations of each value (except zero). This variation is a senders option, and can be used as a broad indication of precision.

Note 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 Appendix III).

10.6 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:

Bits 6 to 1	Number representation
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 future versions of this Recommendation.

Note 1 - The Recommendation in ISO 6093 concerning the user of at least one digit to the left of the decimal mark are also recommended in this Recommendation, but are not mandatory.

Note 2 - There shall be no use of scaling factors specified in accompanying documentation (see ISO 6093).

Note 3 - Use of the normalised form (see ISO 6093) is a senders option, and has no significance.

10.7 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 bit 8 and 7 equal to 0 and 1 respectively are reserved for future versions of this Recommendation.

11 Encoding of a bitstring value

11.1 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.

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

11.2.1 The bits in the bitstring, commencing with the first bit and proceeding to the trailing 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 notation "first bit" and "trailing bit" is specified in Recommendation X.208.

11.2.2 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.

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

11.3 The contents octets for the constructed encoding shall consist of the complete encoding of zero, one or more data values.

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

11.3.1 Each data value encoding in the contents octets shall be the encoding of a value of type BIT STRING.

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

11.3.2 The bits in the bitstring value being encoded, commencing with the first bit and proceeding to the trailing bit, shall be placed in the first bit up to the trailing bit of the first data value encoded in the contents octets, followed by the first bit up to the trailing bit of the second data value encoded in the contents octets, followed by the first bit up to the trailing bit of each data value in turn, followed by the first bit up to the trailing bit of the last data value encoded in the contents octets.

11.3.3 Each data value encoded in the contents octets, with the exception of the last, shall consist of a number of bits which is a multiple of eight.

Note - A data value encoded in the contents octets may be a zero-length bitstring.

11.3.4 Where a constructed encoding is used, there shall be no significance placed on the boundary between the data values encoded in the contents octets.

11.3.5 The encoding of each data value encoded in the contents octets may be primitive or constructed.

Note - It is usually primitive.

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}	$000A3B_{16}$
		BitString	*Length*	*Contents*
		03_{16}	05_{16}	$045F291CD0_{16}$
		EOC	*Length*	
		00_{16}	00_{16}	

12 Encoding of an octetstring value

12.1 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.

12.2 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.

12.3 The contents octets for the constructed encoding shall consist of the complete encoding of zero, one or more data values.

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

12.3.1 Each data value encoding in the contents octets shall be the encoding of a value of type octetstring.

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

12.3.2 The octets in the octetstring value being encoded, commencing with the first octet and proceeding to the trailing octet, shall be placed in the first up to the trailing octet of the first data value encoded in the contents octets, followed by the first up to the trailing octet of the second data value encoded in the contents octets, followed by the first up to the trailing octet of each data value in turn, followed by the first up to the trailing octet of the last data value encoded in the contents octets.

Note - A data value encoded in the contents octets may he a zero length octet string.

12.3.3 here a constructed encoding is used, there shall be no significance placed on the boundary between the data values encoded in the contents octets.

12.3.4 The encoding of each data value encoded in the contents octets may be primitive or constructed.

Note - It is usually primitive.

13 Encoding of a null value

13.1 The encoding of a null value shall be primitive.

13.2 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:

	Null	*Length*
05_{16}	00_{16}	

14 Encoding of a sequence value

14.1 The encoding of a sequence value shall be constructed.

14.2 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".

14.3 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}

15 Encoding of a sequence-of value

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

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

15.3 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.

16 Encoding of a set value

16.1 The encoding of a set value shall be constructed.

16.2 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".

16.3 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.

17 Encoding of a set-of value

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

17.2 The text of § 15.2 applies

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

18 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.

Note 1 - The encoding may be primitive or constructed depending on the chosen type.

Note 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.

19 Encoding of a selection value

The encoding of a selection value shall be the same as the encoding of a value of the selected type.

Note - The encoding may be primitive or constructed depending on the selected type.

20 Encoding of a tagged value

20.1 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 § 20.2 and § 20.3.

20.2 If the "IMPLICIT" keyword was not used in the definition of the type, the encoding shall be constructed and the contents octets shall be the complete base encoding.

20.3 If the "IMPLICIT" keyword 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 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 Typel:

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]	*Length*	*Contents*	
43_{16}	05_{16}	$4A6F6E6573_{16}$	

For Type4:

[Application 7]	*Length*	*Contents*	
67_{16}	07_{16}		
[Application 31	*Length*	*Contents*	
43_{16}	05_{16}	$4A6F6E6573_{16}$	

For Type5:

[2]	*Length*	*Contents*
82_{16}	05_{16}	$4A6F6E6573_{16}$

21 Encoding of a value of the ANY type

The encoding of an ANY type shall be the complete encoding specified in this Recommendation for the type of the value of the ANY type.

22 Encoding of an object identifier value

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

22.2 The contents octets shall be an (ordered) list of encoding of subidentifiers (see § 22.3 and § 22.4) 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 encoded 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 80 (hexadecimal).

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

22.4 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 recognises 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.

22.5 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}

23 Encoding for values of the character string types

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

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

23.3 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 Recommendation X.208. The value of the octet string is specified in §§ 23.4 and 23.5.

23.4 Where a character string type is specified in Recommendation X.208 by direct reference to an enumerating table (NumericString and PrintableString), the value of the octet string shall be that specified in § 23.5 for a VisibleString type with the same character string value.

23.5 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.

23.5.1 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 Recommendation X.208.

TABLE 2/X.209

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	2	None	ESC 2/8 4/0 LS0	NO
PrintableString	2	None	ESC 2/8 4/0 LS0	NO
TeletexString (T61 String)	102	106(C0) 107(C1)	ESC 2/8 7/5 LS0 ESC 2/1 4/5 ESC 2/2 4/8	YES
VideotexString	102	1(C0) 73(C1)	ESC 2/8 7/5 LS0 ESC 2/1 4/0 ESC 2/2 4/1	YES
VisibleString (ISO646String)	2	None	ESC 2/8 4/0 LS0	NO
IA5String	2	1(C0)	ESC 2/8 4/0 LS0 ESC 2/1 4/0	NO
GraphicString	2	None	ESC 2/8 4/0 LS0	YES
GeneralString	2	1(C0)	ESC 2/8 4/0 LS0 ESC 2/1	YES

Note - Many of the commonly used characters (for example, A to 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 § 5.3).

23.5.2 At the start of each string, certain registration numbers shall be assumed to be designated as G0 and/or C0 and/or Cl, and invoked (using the terminology of ISO 2022). These are specified for each type in Table 2/X.209, together with the assumed escape sequence they imply.

23.5.3 Certain character string types shall not contain explicit escape sequences in their encodings; in all other cases, any escape allowed by § 23.5.1 can appear at any time, including at the start of the encoding. Table 2/X.209 lists the types for which explicit escape sequences are allowed.

23.5.4 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.

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}	
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}	
OctetString	*Length*	*Contents*
04_{16}	03_{16}	$4A6F6E_{16}$
OctetString	*Length*	*Contents*
04_{16}	02_{16}	6573_{16}
EOC	*Length*	
00_{16}	00_{16}	

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 § 5.3).

24 Encoding for values of the ASN.1 useful types

A definition of these types using ASN.1 is provided in Recommendation X.208. The encoding shall be that obtained by applying the rules specified in this Recommendation to that type definition.

25 Use in transfer syntax definition

25.1 The encoding rules specified in this Recommendation 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.

25.2 The object identifier and object descriptor values

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

and

"Basic Encoding of a single ASN.1 type"

are assigned to identify and describe the encoding rules specified in this Recommendation.

25.3 Where an application 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 the object identifier value specified in § 25.2 may be used with the abstract syntax name to identify that transfer syntax which results from the application of the encoding rules specified in this Recommendation to the specifically named ASN.1 type used in defining the abstract syntax.

Note - In particular, this identification of the encoding rules can appear in the "transfer syntax name" field of the presentation protocol (Recommendation X.226).

25.4 The name specified in § 25.2 shall not be used with an abstract syntax name to identify a transfer syntax if the conditions of § 25.3 for the definition of the abstract syntax are not met.

APPENDIX I
(to Recommendation X.209)

Example of encodings

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

I.1 *ASN.1 description of the record structure*

The structure of the hypothetical personnel record is formally described below using ASN.1 specified in Recommendation X.208 for defining types.

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

I.2 *ASN.1 description of a record value*

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

```
{                       {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"}}}
```

I.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) 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						
				Visible-String	*Length*	*Contents*				
				1A	04	"John"				
				Visible-String	*Length*	*Contents*				
				1A	01	"P"				
				Visible-String	*Length*	*Contents*				
				1A	05	"Smith"				
		Title	*Length*	*Contents*						
		A0	0A							
				Visible-String	*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					
						Visible-String	*Length*	*Contents*		
						1A	04	"Mary"		
						Visible-String	*Length*	*Contents*		
						1A	01	"T"		
						Visible-String	*Length*	*Contents*		
						1A	05	"Smith"		
		[3]	*Length*	*Contents*						
		A3	42							
				Set	*Length*	*Contents*				
				31	IF					
						Name	*Length*	*Contents*		
						61	11			
								Visible-String	*Length*	*Contents*
								1A	05	"Ralph"
								Visible-String	*Length*	*Contents*
								1A	01	"T"
								Visible-String	*Length*	*Contents*
								1A	05	"Smith"
						Date of Birth	*Length*	*Contents*		
						A0	0A			
								Date	*Length*	*Contents*
								43	08	"19571111"
				Set	*Length*	*Contents*				
				31	IF					
						Name	*Length*	*Contents*		
						61	11			
								Visible-String	*Length*	*Contents*
								16	05	"Susan"
								Visible-String	*Length*	*Contents*
								16	01	"B"
								Visible-String	*Length*	*Contents*
								1	05	"Jones"
								Date of Birth	*Length*	*Contents*
								A0	0A	
								Date	*Length*	*Contents*
								43	08	"19590717"

APPENDIX II

(to Recommendation X.209)

Assignment of object identifier values

The following values are assigned in this Recommendation:

Clause	**Object Identifier Value** **Object Descriptor Value**
25.2	{joint-iso-ccitt asnl (1) basic-encoding (1)} "Basic Encoding of a single ASN.1 type"

APPENDIX III

(to Recommendation X.209)

Illustration of real value encoding

III.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 Appendix illustrates the steps which would be taken in such a process by using the (artificial) hardware floating point representation of the mantissa shown in Figure III-I/X.209.

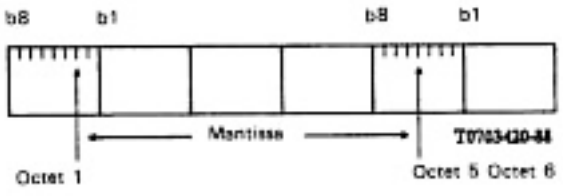

FIGURE III -1/X.209

Hypothetical hardware representation

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

III.2 The contents octets which need to be generated for sending a non-zero value (as specified in the body of this Recommendation) are:

1 S bb ff ee Octets for E 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 § III.3, and ee is a fixed length of exponent value calculated as described in § III.4 (this Appendix does not treat the case where E needs to exceed three octets).

III.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 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 6 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 N, the octets transferred, is after bit 1 of octet 5.) Thus we have the crucial parameters:

F = 3 (so ff is 11)

exponent decrement = 9

111.4 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 § III.3. Let us assume this gives a length of 3 octets. Then ee is 10. Let us also assume excess is zero.

III.5 The transmission algorithm is now:

a) test for zero, and if so, transmit an ASN.1 length of zero (no contents octets) and end the algorithm;

b) test and remember the mantissa sign, and negate the mantissa if negative;

c) transmit an ASN.1 length of 9, then

11101110 if negative

or

10101110 if positive

d) produce and transmit the 3 octet exponent with value

E - 9

e) zero bits 8 to 3 of octet 1 and bits 4 to 1 of octet 5, then transmit the 5 octet mantissa.

III.6 The receiving algorithm has to be prepared to handle any ASN.1 format, 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 optimisation possible in special cases, but may find (apart from the optimisation relating to transmissions from a compatable machine) that testing for them loses more than they gain.

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