INTERNATIONAL STANDARD

ISO 8632-3

First edition 1987-08-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ

Information processing systems — Computer graphics — Metafile for the storage and transfer of picture description information —

Part 3 :
Binary encoding

Systèmes de traitement de l'information — Infographie — Métafichier de stockage et de transfert des informations de description d'images —

Partie 3 : Codage binaire

Reference number
ISO 8632-3 : 1987 (E)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work.

Draft International Standards adopted by the technical committees are circulated to the member bodies for approval before their acceptance as International Standards by the ISO Council. They are approved in accordance with ISO procedures requiring at least 75 % approval by the member bodies voting.

International Standard ISO 8632-3 was prepared by Technical Committee ISO/TC 97, *Information processing systems*.

Users should note that all International Standards undergo revision from time to time and that any reference made herein to any other International Standard implies its latest edition, unless otherwise stated.

CONTENTS

Information processing systems — Computer graphics — Metafile for the storage and transfer of picture description information —

Part 3 : Binary encoding

0 Introduction

0.1 Purpose of the Binary Encoding

The Binary Encoding of the Computer Graphics Metafile (CGM) provides a representation of the Metafile syntax that can be optimized for speed of generation and interpretation, while still providing a standard means of interchange among computer systems. The encoding uses binary data formats that are much more similar to the data representations used within computer systems than the data formats of the other encodings.

Some of the data formats may exactly match those of some computer systems. In such cases processing is reduced very much relative to the other standardized encodings. On most computer systems processing requirements for the Binary Encoding will be substantially lower than for the other encodings.

In cases where a computer system's architecture does not match the standard formats used in the Binary Encoding, and where absolute minimization of processing requirements is critical, and where interchange among dissimilar systems does not matter, it may be more appropriate to use a private encoding, conforming to the rules specified in clause 7 of ISO 8632/1.

0.2 Objectives

This encoding has the following features.

a) Partitioning of parameter lists: metafile elements are coded in the Binary Encoding by one or more partitions (see clause 4); the first (or only) partition of an element contains the opcode (Element Class plus Element Id).

b) Alignment of elements: every element begins on a word boundary. Alignment of partitions that require an odd number of octets is effected by padding with an octet with all bits zero. A no-op element is available in this encoding; it is ignored. It may be used to align data on machine-dependent record boundaries for speed of processing.

c) Uniformity of format: all elements have an associated parameter length value. The length is specified as an octet count. As a result, it is possible to scan the metafile, without interpreting it, at high speed.

d) Alignment of coordinate data: at default precisions and by virtue of alignment of elements, coordinate data always start on word boundaries. This minimizes processing by ensuring, on a wide class of computing systems, that single coordinates do not have to be assembled from pieces of multiple computer words.

e) Efficiency of encoding integer data: other data such as indexes, colour and characters are encoded as one or more octets. The precision of every parameter is determined by the appropriate precision as given in the METAFILE DESCRIPTOR.

f) Order of bit data: in each word, or unit within a word, the bit with the highest number is the most significant bit. Likewise, when data word are accessed sequentially, the least significant word follows the most significant.

g) Extensibility: the arrangement of Element Class and Element Id values has been designed to allow future growth, such as new graphical elements.

h) Format of real data: real numbers are encoded using either IEEE floating point representation or a metafile fixed-point representation.

i) Run length encoding: if many adjacent cells have the same colour (or colour index) efficient encoding is possible. For each run, the colour (or colour index) is specified, followed by a cell count.

j) Packed list encoding: if adjacent colour cells do not have the same colour (or colour index) the metafile provides bit-stream lists in which the values are packed as closely as possible.

0.3 Relationship to other International Standards

The floating point representation of real data in this part of ISO 8632 is that in ANSI/IEEE 754-1986.

The representation of character data in this part of ISO 8632 follows the rules of ISO 646 and ISO 2022.

For certain elements, the CGM defines value ranges as being reserved for registration. The values and their meanings will be defined using the established procedures (see ISO 8632/1, 4.11.)

0.4 Status of annexes

The annexes do not form an integral part of this part of ISO 8632 but are included for information only.

1 Scope and field of application

This part of ISO 8632 specifies a binary encoding of the Computer Graphics Metafile. For each of the elements specified in ISO 8632/1, an encoding is specified in terms of a data type. For each of these data types, an explicit representation in terms of bits, octets and words is specified. For some data types, the exact representation is a function of the precisions being used in the metafile, as recorded in the METAFILE DESCRIPTOR.

This encoding of the Computer Graphics Metafile will, in many circumstances, minimize the effort required to generate and interpret the metafile.

2 References

ISO 646, *Information Processing—ISO 7-bit coded character set for information interchange.*

ISO 2022, *Information Processing—ISO 7-bit and 8-bit coded character sets—Code extension techniques.*

ANSI/IEEE 754, *Standard for Binary Floating Point Arithmetic.*

3 Notational conventions

"Command Header" is used throughout this part to refer to that portion of a Binary-Encoded element that contains the opcode (element class plus element id) and parameter length information (see clause 4).

Within this part, the terms "octet" and "word" have specific meanings. These meanings may not match those of a particular computer system on which this encoding of the metafile is used.

An octet is an 8-bit entity. All bits are significant. The bits are numbered from 7 (most significant) to 0 (least significant).

A word is a 16-bit entity. All bits are significant. The bits are numbered from 15 (most significant) to 0 (least significant).

4 Overall structure

4.1 General form of metafile

All elements in the metafile are encoded using a uniform scheme. The elements are represented as variable length data structures, each consisting of opcode information (element class plus element id) designating the particular element, the length of its parameter data and finally the parameter data (if any).

The structure of the metafile is as follows. (For the purposes of this diagram only, MF is used as an abbreviation for METAFILE.)

BEGIN MF	MD	<picture> ...	END MF

The BEGIN METAFILE element is followed by the METAFILE DESCRIPTOR (MD). After this the pictures follow, each logically independent of each other. Finally the Metafile is ended with an END METAFILE element.

4.2 General form of pictures

Apart from the BEGIN METAFILE, END METAFILE and Metafile Descriptor elements, the metafile is partitioned into pictures. All pictures are mutually independent. A picture consists of a BEGIN PICTURE element, a PICTURE DESCRIPTOR (PD) element, a BEGIN PICTURE BODY element, an arbitrary number of control, graphical and attribute elements and finally an END PICTURE element. (For the purpose of this diagram only, PIC is used as an abbreviation for PICTURE and BEGIN BODY for BEGIN PICTURE BODY.)

BEGIN PIC	PD	BEGIN BODY	<element> ...	END PIC

4.3 General structure of the binary metafile

The binary encoding of the metafile is a logical data structure consisting of a sequential collection of bits. For convenience in describing the length and alignment of metafile elements, fields of two different sizes are defined within the structure. These fields are used in the remainder of ISO 8632/3 for illustrating the contents and structure of elements and parameters.

For measuring the lengths of elements the metafile is partitioned into octets, which are 8-bit fields.

The structure is also partitioned into 16-bit fields called words (these are logical metafile words). To optimize processing of the binary metafile on a wide collection of computers, metafile elements are constrained to start on word boundaries within the binary data structure (this alignment may necessitate padding an element with bits to a word boundary if the parameter data of the element does not fill to such a boundary).

The octet is the fundamental unit of organization of the binary metafile.

The bits of an octet are numbered 7 to 0, with 7 being the most significant bit. The bits of a word are numbered 15 to 0, with 15 being the most significant bit.

```
              b7              b0
              +-+-+-+-+-+-+-+-+
    octet:    |               |
              +-+-+-+-+-+-+-+-+
             msb             lsb

             b15             b8.b7             b0
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    word:     |               |               |
              +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
             msb                             lsb
```

If the consecutive bits of the binary data structure are numbered 1..N, and the consecutive octets are numbered 1..N/8, and the consecutive words are numbered 1..N/16, then the logical correspondence of bits, octets, and words in the binary data structure is as illustrated in the following table:

metafile bit number	octet bit number	word bit number
1	b7/octet1	b15/word1
.	.	.
.	.	.
8	b0/octet1	b8/word1
9	b7/octet2	b7/word1
.	.	.
.	.	.
16	b0/octet2	b0/word1
17	b7/octet3	b15/word2
.	.	.
.	.	.
24	b0/octet3	b8/word2
25	b7/octet4	b7/word2
.	.	.
.	.	.

4.4 Structure of the command header

Throughout this sub-clause, the term "command" is used to denote a binary-encoded element. Metafile elements are represented in the Binary Encoding in one of two forms — short-form commands and long-form commands. There are two differences between them:

— a short-form command always contains a complete element; the long-form command can accommodate partial elements (the data lists of elements can be partitioned);

— a short-form command only accommodates parameter lists up to 30 octets in length; the long-form command accommodates lengths up to 32767 octets per data partition.

The forms differ in the format of the Command Header that precedes the parameter list. The command form for an element (short or long) is established by the first word of the element. For the short-form, the Command Header consists of a single word divided into three fields: element class, element id and parameter list length.

```
            15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
           ______________________________________________________________
  Word 1   | elem class |      element  id          | parm  list  len |
           |____________|___________________________|_________________|
```

Figure 1 — Format of a short-form Command Header.

Structure of the command header

The fields in the short-form Command Header are as follows.

bits 15 to 12	element class (value range 0 to 15)
bits 11 to 5	element id (value range 0 to 127)
bits 4 to 0	parameter list length: the number of octets of parameter data that follow for this command (value range 0 to 30)

This Command Header is then followed by the parameter list.

The first word of a long-form command is identical in structure to the first word of a short-form command. The presence of the value 11111 binary (decimal 31) in parameter list length field indicates that the command is a long-form command. The Command Header for the long-form command consists of two words. The second word contains the actual parameter list length. The two header words are then followed by the parameter list.

In addition to allowing longer parameter lists, the long-form command allows the parameter list to be partitioned. Bit 15 of the second word indicates whether the given data complete the element or more data follow. For subsequent data partitions of the element, the first word of the long-form Command Header (containing element class and element id) is omitted; only the second word, containing the parameter list length, is given. The parameter list length for each partition specifies the length of that partition, not the length of the complete element. The final partition of an element is indicated by bit 15 of the parameter list length word being zero.

```
           15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          ------------------------------------------------
Word  1   | elem class |      element  id     | 1  1  1  1  1|
          |------------|----------------------|--------------|
Word  2   | P |         parameter list length               |
          |---|---------------------------------------------|
```

Figure 2 — Format of a long-form Command Header.

The fields in the long-form Command Header are as follows.

Word 1

bits 15 to 12	element class (value range 0 to 15)
bits 11 to 5	element id (value range 0 to 127)
bits 4 to 0	binary value 11111 (decimal 31) indicating long-form

Word 2

bit 15 partition flag

— 0 for 'last' partition

— 1 for 'not-last' partition

bits 14 to 0 parameter list length: the number of octets of parameter data that follow for this command or partition (value range 0 to 32767).

The parameter values follow the parameter list length for either the long-form or short-form commands. The number of values is determined from the parameter list length and the type and precision of the operands. These parameter values have the format illustrated in clause 5 of this part. The parameter type for coordinates is indicated in the Metafile Descriptor. For non-coordinate parameters, the parameter type is as specified in clause 5 of part 1. If the parameter type is encoding dependent, its code is specified in the coding tables of clause 7 of this part. Unless otherwise stated, the order of parameters is as listed in clause 5 of part 1.

Every command is constrained to begin on a word boundary. This necessitates padding the command with a single null octet at the end of the command if the command contains an odd number of octets of parameter data. In addition, in elements with parameters whose precisions are shorter than one octet (i.e., those containing a 'local colour precision' parameter) it is necessary to pad the last data-containing octet with null bits if the data do not fill the octet. In all cases, the parameter list length is the count of octets actually containing parameter data — it does not include the padding octet if one is present. It is only at the end of a command that padding is performed, with the single exception of the CELL ARRAY element.

The purpose of this command alignment constraint is to optimize processing on a wide class of computers. At the default metafile precisions, the parameters which are expected to occur in greatest numbers (coordinates, etc) will align on 16-bit boundaries, and Command Headers will align on 16-bit boundaries. Thus, at the default precisions the most frequently parsed entities will lie entirely within machine words in a large number of computer designs. The avoidance of assembling single metafile parameters from pieces of several computer words will approximately halve the amount of processing required to recover element parameters and command header fields from a binary metafile data stream.

This optimization may be compromised or destroyed altogether if the metafile precisions are changed from default. Commands are still constrained to begin on 16-bit boundaries, but the most frequently expected parameters may no longer align on such boundaries as they do at the default precisions.

The short form command header with element class 15, element id 127, and parameter list length 0 is reserved for extension of the number of available element classes in future revisions of ISO 8632/3. It should be treated by interpreters as any other element, as far as parsing is concerned. The next "normal" element encountered will have an actual class value different from that encountered in the "element class" field of the command header — it will be adjusted by a bias as will be defined in a future revision of ISO 8632/3.

5 Primitive data forms

The Binary Encoding of the CGM uses five primitive data forms to represent the various abstract data types used to describe parameters in ISO 8632/1.

The primitive data forms and the symbols used to represent them are as follows.

SI	Signed Integer
UI	Unsigned Integer
C	Character
FX	Fixed Point Real
FP	Floating Point Real

Each of these primitive forms (except Character) can be used in a number of precisions. The definitions of the primitive data forms in sub-clauses 5.1 to 5.5 show the allowed precisions for each primitive data form. The definitions are in terms of 'metafile words' which are 16-bit units.

The following terms are used in the following diagrams when displaying the form of numeric values.

msb	most significant bit
lsb	least significant bit
S	sign bit

The data types in the following data diagrams are illustrated for the case that the parameter begins on a metafile word boundary. In general, parameters may align on odd or even octet boundaries, because they may be preceded by an odd or even number of octets of other parameter data. Elements containing the local colour precision parameter may have parameters shorter than one octet. It is possible in such cases that the parameters will not align on octet boundaries.

5.1 Signed integer

Signed integers are represented in "two's complement" format. Four precisions may be specified for signed integers: 8-bit, 16-bit, 24-bit and 32-bit. (Integer coordinate data encoded with this primitive data form do not use the 8-bit precision.)

5.1.1 Signed integer at 8-bit precision

Each value occupies half a metafile word (one octet).

```
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
-----------------------------------------------
|S|msb    value  i     lsb|S|msb    value  i+1   lsb|
|-|-----------------------|-|-----------------------|
```

5.1.2 Signed integer at 16-bit precision

Each value occupies one metafile word.

```
15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
-----------------------------------------------
| S |msb              value                 lsb|
|---|-------------------------------------------|
```

5.1.3 Signed integer at 24-bit precision

Each value straddles two successive metafile words.

```
          15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          ------------------------------------------------
Word 1   | S|msb                value i
         |--|---------------------------------------------
Word 2          value i           lsb|S|msb   value i+1
          ---------------------------|-|------------------
Word 3                          value i+1               lsb|
          -------------------------------------------------|
```

5.1.4 Signed integer at 32-bit precision

Each value fills two complete metafile words.

```
          15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          ------------------------------------------------
Word 1   | S|msb                value i
         |--|---------------------------------------------
Word 2                          value i                 lsb|
         --------------------------------------------------|
```

5.2 Unsigned integer

Four precisions may be specified for unsigned integers: 8-bit, 16-bit, 24-bit and 32-bit.

5.2.1 Unsigned integers at 8-bit precision

Each value occupies half a metafile word.

```
          15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          ------------------------------------------------
         |msb      value i      lsb|msb    value i+1    lsb|
         |-------------------------|-----------------------|
```

5.2.2 Unsigned integers at 16-bit precision

Each value occupies one metafile word.

```
          15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          ------------------------------------------------
         |msb                    value                  lsb|
         |-------------------------------------------------|
```

5.2.3 Unsigned integers at 24-bit precision

Each value straddles two successive metafile words.

```
          15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          ------------------------------------------------
Word 1   |msb                   value i
         |------------------------------------------------
Word 2           value i          lsb|msb      value i+1
          ---------------------------|--------------------
Word 3                          value i+1               lsb|
          -------------------------------------------------|
```

5.2.4 Unsigned integers at 32-bit precision

Each value fills two complete metafile words.

```
          15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          -----------------------------------------------
Word 1   |msb                value i
         |-----------------------------------------------
Word 2                       value i                  lsb|
          -----------------------------------------------|
```

5.3 Character

Each character is stored in an octet.

```
          15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          -----------------------------------------------
         |      Character i        |    Character i+1    |
         |-------------------------|---------------------|
```

5.4 Fixed point real

Fixed point real values are stored as two integers; the first represents the "whole part" and has the same form as a Signed Integer (SI; see 5.1); the second represents the "fractional part" and has the same form as an Unsigned Integer (UI; see sub-clause 5.2). Two precisions may be specified for Fixed Point Reals: 32-bit or 64-bit.

5.4.1 Fixed point real at 32-bit precision

Each Fixed Point Real occupies 2 complete metafile words; the first has the form of a 16-bit Signed Integer and the second the form of a 16-bit Unsigned Integer.

```
          15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          -----------------------------------------------
Word 1   | S |msb        Whole part                   lsb|
         |---|-------------------------------------------|
Word 2   |msb            Fraction part                lsb|
         |-----------------------------------------------|
```

5.4.2 Fixed point real at 64-bit precision

Each Fixed Point Real occupies 4 complete metafile words; the first has the form of a 32-bit Signed Integer and the second the form of a 32-bit Unsigned Integer.

```
          15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
          -----------------------------------------------
Word 1   | S | msb       whole part
         |---|-------------------------------------------
Word 2                   whole part                  lsb |
          -----------------------------------------------|
Word 3   | msb           fraction part
         |-----------------------------------------------
Word 4                   fraction part               lsb |
          -----------------------------------------------|
```

5.4.3 Value of fixed point reals

The values of the represented real numbers are given by:

for 32 bits: $$\text{real\_value} = SI + \left(\frac{UI}{2^{16}}\right)$$

for 64 bits: $$\text{real\_value} = SI + \left(\frac{UI}{2^{32}}\right)$$

SI stands for the "whole part" and UI stands for the "fractional part" in these equations. SI, the whole part, is the largest integer less than or equal to the real number being represented.

5.5 Floating point

Floating Point Real values are represented in the floating point format of ANSI/IEEE 754. This format contains three parts:

— a sign bit ('s');

— a biased exponent part ('e');

— a fraction part ('f').

The value is a function of these three values ('s', 'e' and 'f'). If 's' is '0', the value is positive; if 's' is '1', the value is negative. Two precisions may be specified for Floating Point Reals: 32-bit or 64-bit. The magnitude of the value is calculated as follows for 32-bit representation.

a) If $e = 255$ and $f \neq 0$, then the value is undefined.

b) If $e = 255$ and $f = 0$, then the value is as large a positive (s=0) or negative (s=1) value as possible.

c) If $0 < e < 255$, then the magnitude of the value is $(1.f)(2^{e-127})$.

d) If $e = 0$ and $f \neq 0$, then the magnitude of the value is $(0.f)(2^{-126})$.

e) If $e = 0$ and $f = 0$, then the value is 0.

The magnitude of the value is calculated as follows for 64-bit representation.

a) If $e = 2047$ and $f \neq 0$, then the value is undefined.

b) If $e = 2047$ and $f = 0$, then the value is as large a positive (s=0) or negative (s=1) value as possible.

c) If $0 < e < 2047$, then the magnitude of the value is $(1.f)(2^{e-1023})$.

d) If $e = 0$ and $f \neq 0$, then the magnitude of the value is $(0.f)(2^{-1022})$.

e) If $e = 0$ and $f = 0$, then the value is 0.

5.5.1 Floating point real at 32-bit precision

Each Floating Point Real value occupies 2 metafile words. The size of each field in the value is as follows.

sign	1 bit
exponent	8 bits
fraction	23 bits

```
            15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
            -----------------------------------------------
  Word 1   | S |msb      Exponent       lsb|msb   Fraction
           |---|---------------------------|---------------
  Word 2                   Fraction                      lsb|
            -----------------------------------------------|
```

5.5.2 Floating point real at 64-bit precision

Each Floating Point Real value occupies 4 metafile words. The size of each field in the value is as follows.

sign	1 bit
exponent	11 bits
fraction	52 bits

```
            15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
            -----------------------------------------------
  Word 1   | S |msb        Exponent             lsb|msb
           |---|-----------------------------------|--------
  Word 2                   Fraction
            -----------------------------------------------
  Word 3                   Fraction
            -----------------------------------------------
  Word 4                   Fraction                      lsb|
            -----------------------------------------------|
```

6 Representation of abstract parameter types

Table 1 shows, for each of the abstract parameter types, how it is represented in the Binary Encoding of the CGM in terms of primitive data forms. The columns of the table are as follows.

1) The symbol for the abstract parameter type, as it is specified in clause 5 of ISO 8632/1.

2) The way the parameter type is constructed in terms of the primitive data forms, at the appropriate precisions. The precisions are those defined in clause 5 of ISO 8632/1.

3) The symbol for the number of octets required to represent one instance (occurance) of the given parameter, at the given precision, and the formula for computing the number.

4) The symbol for the range of values which the parameter can assume, followed by the numerical values which the parameter can assume, followed by the numerical values which define the range.

The symbols of columns 3 and 4 are used extensively in the code tables in clause 7. Also used in the code tables are variations on those symbols:

+IR, +RR, ..	denote the range of positive integers, range of positive reals, ..
++IR, ++RR, ..	denote the range of non-negative integers, range of non-negative reals, ..
mI, mR ..	denotes 'm' integers, reals, ..
I*, R* ..	denotes an unbounded number of integers, reals, ..

Combinations are used:

2R, 2I, IX*	indicates a parameter that is represented by 2 reals, then a parameter that is represented by 2 integers and finally a parameter that contains an unlimited number of index values.

Representation of abstract parameter types

Table 1 — Representation of abstract data types.

Abstract symbol	Parameter construction from primitive forms	Octets per parameter : symbol and value	Parameter range: symbol and value
CI	UI at colour index precision (cip)	BCI {=cip/8}	CIR {0..(2**cip-1)}
CD	3UI at direct colour precision (dcp)	BCD {=3*dcp/8} {see notes 1,2}	CDR {0..(2**dcp-1)}
IX	SI at index precision (ixp)	BIX {=ixp/8}	IXR {-2**(ixp-1) to 2**(ixp-1)-1 }
E	SI at fixed precision (16-bit) {see note 3}	BE {=2}	IXR {-2**(15) to 2**(15)-1 }
I	SI at integer precision (ip)	BI {=ip/8}	IR {-2**(ip-1) to 2**(ip-1)-1 }
R	FP or FX at real precision (rp)	BR {=sum(rp)/8} {see note 4}	RR {=FPR or FXR, see notes 5,10}
S,D	UI,nC	BS {=n+1 or n+3}	SR {see notes 6,12}
VDC	SI at VDC integer precision (vip) or FP or FX at VDC real precision (vrp)	BVDC {=vip/8} or BVDC {=sum(vrp)/8} {see note 4}	VDCR {-2**(vip-1) to 2**(vip-1)-1 } or VDCR {see notes 1,5,7,8}
P	(VDC,VDC)	BP {=2*BVDC}	VDCR {see notes 1,5,7,8}
CO	CI or CD	BCO {=BCI} or BCO {=BCD}	COR {=CIR} {see notes 9,11} or COR {=CDR}

Additional description ("notes") for table 1:

1) For parameters that are composed of multiple identical components (e.g., DIRECT COLOUR, CD, and POINT, P) the range value represents the range of a single component.

2) Direct colour is abstractly a real in the range {0,1}. This is normalized onto the unsigned range CDR in the table.

3) Abstract parameter type Enumeration, E, is encoded identically to abstract type Index, IX, at 16-bit precision.

4) The REAL PRECISION element contains an indicator (fixed or floating point) and two precision components. The symbol "sum(rp)" in the table indicates the sum of the number of bits specified in the two components. The same considerations apply to the VDC REAL PRECISION element and the symbol "sum(vrp)" in the tables. The VDC REAL PRECISION control element may cause 'vrp' to be updated in the body of metafile.

5) FPR and VDCR (when VDC are floating point reals) are computed following the ANSI/IEEE 754 floating point standard (see clause 5 on the Floating Point Data Form).

6) The range for parameter type S is not applicable. The range for character data is not applicable. A string is encoded as a count (unsigned integer) followed by characters. The encoding of

the count is similar to the encoding of length information for metafile commands themselves. If the first octet is in the range 0..254, then it represents the character count for the complete string. If the first octet is 255, then the next 16 bits contain the character count and a continuation flag. The first bit is used as a continuation flag (allowing strings longer than 32767 characters) and the next 15 bits represent the count, 0..32767, for the partial string. If the first bit is 0, then this partial string completes the string parameter. If 1, then this partial string will be followed by another. See CHARACTER SET LIST element.

7) The abstract parameter type VDC, a single VDC value, is either a real or an integer, depending on the declaration of the metafile descriptor function VDC TYPE. Subsequent tables use a single set of symbols, VDC, BVDC and VDCR, recognizing that they are computed differently depending on VDC TYPE.

8) The abstract parameter type VDC is a single value; a point, P, is an ordered pair of VDC.

9) The parameter type symbol CO does not correspond to an abstract parameter type as used in ISO 8632/1. Rather, it is a convenient shorthand for 'colour', which is either direct colour (CD) or indexed colour (CI), depending on the value specified in the COLOUR SELECTION MODE element. The associated octets per parameter and range symbols, BCO and COR, are thus either BCI and CIR or BCD and CDR respectively depending upon COLOUR SELECTION MODE.

10) To eliminate the need to support IEEE floating point in applications that do not need the dynamic range for parameters of type R and VDC, a fixed point real format is provided for scalars (such as line width, character spacing) and VDC. Fixed point reals consist of a (SI,UI) pair.

Fixed point reals (FX) apply to VDC, and to R parameters for the following elements:

a) line width;

b) edge width;

c) character spacing;

d) character expansion factor;

e) marker size;

f) vertical continuous text alignment;

g) horizontal continuous text alignment.

11) CELL ARRAY colour can optionally specify 1, 2, 4, 8, 16, 24 or 32 bit precisions for cell colours, as well as using the default CI or CD precision.

The way in which the colour values in CELL ARRAY is represented is an extension of the representation of single colour values. The CELL ARRAY element has a 'cell representation flag' which may take one of two values:

0	run length representation
1	packed representation

For PACKED mode, each row of the cell array is represented by an array of colour values without compression. Each row starts on a word boundary. No row length information is stored since all rows are the same length.

The colour data thus occupies $2n_y(1 + [pn_x/16])$ octets, where n_x is the number of cells per row, n_y is the number of rows, p is the number of bits per colour, and [..] denotes "the greatest integer in .."

For RUN LENGTH encoding, the data for each row begins on a word boundary and consists of run-length-lists for runs of constant colour value. Each 'run-length-list' consists of a count of a number of consecutive cells and the representation of that colour. In terms of the abstract terms above, the colour list is of format <I,CO>* and its length is <BI,BCO>*.

12) Abstract parameter type Data Record, D, is encoded in this part as string data, S. The coding tables in clause 7 will use the symbol D for the parameter type, and will use the S-related symbols for other data about the parameter.

7 Representation of each element

7.1 Method of presentation

The elements are grouped according to their class; there are eight classes.

Table 2 — List of element class codes

Class	Type of Elements
0	Delimiter elements
1	Metafile Descriptor elements
2	Picture Descriptor elements
3	Control elements
4	Graphical Primitive elements
5	Attribute elements
6	Escape element
7	External elements
8-15	Reserved for future standardization

A complete list of element id codes and element class codes is given in annex C.

For each class this clause contains a sub-clause which consists of a table and a set of notes. The table specifies the metafile element, element id, parameter type, parameter list length, parameter range and default values. The parameter list length is given in octets, which in some cases is constant and in other cases is variable. Any element that contains an odd number of octets is padded with one octet of all zero bits. The parameter list length does not include this extra octet (see 4.4). Following each table there is a set of notes that provide additional details on the coding of each element.

7.2 Delimiter elements

Table 3 — Encoding of delimiter elements.

Element Class 0	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
no-op	0	see below	n	n/a	n/a
BEGIN METAFILE	1	S	BS	SR	null
END METAFILE	2	n/a	0	n/a	n/a
BEGIN PICTURE	3	S	BS	SR	null
BEGIN PICTURE BODY	4	n/a	0	n/a	n/a
END PICTURE	5	n/a	0	n/a	n/a

Additional description of the elements in table 3:

Code Description

0 no-op: has 1 parameter:

P1: an arbitrary sequence of n octets, n=0,1,2..

The parameter, unlike all other parameters in the binary encoding, is not constructed from the primitive data forms — it is an arbitrary sequence of zero or more octets for padding purposes.

1 BEGIN METAFILE: has 1 parameter:

P1: (string) metafile name

2 END METAFILE: has no parameters.

3 BEGIN PICTURE: has 1 parameter:

P1: (string) picture name

4 BEGIN PICTURE BODY: has no parameters.

5 END PICTURE: has no parameters.

7.3 Metafile descriptor elements

Table 4 — Encoding of metafile descriptor elements.

Element Class 1	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
METAFILE VERSION	1	I	BI	+IR(1..n)	see below
METAFILE DESCRIPTION	2	S	BS	SR	null
VDC TYPE	3	E	BE	0,1	0
INTEGER PRECISION	4	I	BI	8,16,24,32	16
REAL PRECISION	5	E,2I	BE+2BI	{0,1}, {9,12,16,32}, {23,52,16,32}	1,16,16
INDEX PRECISION	6	I	BI	8,16,24,32	16
COLOUR PRECISION	7	I	BI	8,16,24,32	8
COLOUR INDEX PRECISION	8	I	BI	8,16,24,32	8
MAXIMUM COLOUR INDEX	9	CI	BCI	CIR	63
COLOUR VALUE EXTENT	10	2CD	2BCD	CDR	see below
METAFILE ELEMENT LIST	11	I,2nIX	BI,2nBIX	++IR,IXR	n/a
METAFILE DEFAULTS REPLACEMENT	12	Metafile elements	variable	Metafile elements	none
FONT LIST	13	nS	nBS	SR	see below
CHARACTER SET LIST	14	n(E,S)	n(BE+BS)	{0..4},SR	see below
CHARACTER CODING ANNOUNCER	15	E	BE	0,1,2,3	0

Additional description of the elements in table 4:

Code Description

1 METAFILE VERSION: has 1 parameter:

P1: (integer) metafile version number; for this metafile, always 1

2 METAFILE DESCRIPTION: has 1 parameter:

P1: (string) metafile descriptive string

3 VDC TYPE: has 1 parameter:

P1: (enumerated) VDC TYPE: valid values are:

0	VDC values specified in integers
1	VDC values specified in reals

4 INTEGER PRECISION: has 1 parameter:

P1: (integer) integer precision: 8, 16, 24 or 32 are the only valid values

5 REAL PRECISION: has 3 parameters:

P1: (enumerated) form of representation for real values: valid values are:

0	floating point format
1	fixed point format

P2: (integer) field width for exponent or whole part (including 1 bit for sign)
P3: (integer) field width for fraction or fractional part

Legal combinations of values are:

P1	P2	P3	Result
0	9	23	32-bit floating point
0	12	52	64-bit floating point
1	16	16	32-bit fixed point
1	32	32	64-bit fixed point

6 INDEX PRECISION: has 1 parameter:

P1: (integer) Index precision: valid values are 8,16,24,32

7 COLOUR PRECISION: has 1 parameter:

P1: (integer) Colour precision: valid values are 8,16,24,32

8 COLOUR INDEX PRECISION: has 1 parameter:

P1: (integer) Colour index precision: valid values are 8,16,24,32

9 MAXIMUM COLOUR INDEX: has 1 parameter:

P1: (colour index) maximum colour index that may be encountered in the metafile.

10 COLOUR VALUE EXTENT: has 2 parameters:

P1: (direct colour value) minimum colour value
P2: (direct colour value) maximum colour value

The defaults for the two parameters are (0,0,0) and (255,255,255), respectively.

11 METAFILE ELEMENTS LIST: has 2 parameters:

P1: (integer) number of elements specified
P2: (index-pair array) List of metafile elements in this metafile. Each element is represented by two values: the first is its element class code (as in table 2) and the second is its element id code (as in tables 3 to 10). These codes are listed in annex C. The shorthand pseudo-elements are represented by:

drawing set:	(-1,0)
drawing-plus-control set:	(-1,1)

12 METAFILE DEFAULTS REPLACEMENT: has 1 parameter that itself contains metafile elements. The structure and format is identical to appropriate metafile element(s).

13 FONT LIST: has a variable parameter list:

P1-Pn: n font names (strings)

The default font list consists of one entry: any font capable of representing the standard national character set based on ISO 646.

14 CHARACTER SET LIST: has a variable number of parameter pairs; for each of these:

P1: (enumerated) CHARACTER SET TYPE: valid codes are:

0	94-character G-set
1	96-character G-set
2	94-character multibyte G-set
3	96-character multibyte G-set
4	complete code

negative for private use

P2: (string) Designation sequence tail; see part 1, 5.3.13.

The default character set list consists of one entry: the value of the enumerated parameter is 0, and the value of the string parameter is the designation sequence tail that is registered for the standard national character set based on ISO 646.

15 CHARACTER CODING ANNOUNCER: has 1 parameter:

P1: (enumerated) character coding announcer: valid values are:

0	basic 7-bit
1	basic 8-bit
2	extended 7-bit
3	extended 8-bit

negative for private use

7.4 Picture descriptor elements

Table 5 — Encoding of picture descriptor elements.

Element Class 2	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
SCALING MODE	1	E,R (FP)	BE+BFP	{0,1},FPR	0,-
COLOUR SELECTION MODE	2	E	BE	{0,1}	0
LINE WIDTH SPECIFICATION MODE	3	E	BE	{0,1}	1
MARKER SIZE SPECIFICATION MODE	4	E	BE	{0,1}	1
EDGE WIDTH SPECIFICATION MODE	5	E	BE	{0,1}	1
VDC EXTENT	6	2P	2BP	VDCR	see below
BACKGROUND COLOUR	7	CD	BCD	CDR	see below

Additional description of the elements in table 5:

Code Description

1 SCALING MODE: has 2 parameters:

P1: (enumerated) scaling mode: valid values are:

0	abstract scaling
1	metric scaling

P2: (real) metric scaling factor, ignored if P1=0

2 COLOUR SELECTION MODE: has 1 parameter:

P1: (enumerated) colour selection mode:

0	indexed colour mode
1	direct colour mode

3 LINE WIDTH SPECIFICATION MODE: has 1 parameter:

P1: (enumerated) line width specification mode: valid values are:

0	absolute
1	scaled

4 MARKER SIZE SPECIFICATION MODE: has 1 parameter:

P1: (enumerated) marker size specification mode: valid values are:

0	absolute
1	scaled

5 EDGE WIDTH SPECIFICATION MODE: has 1 parameter:

P1: (enumerated) edge width specification mode: valid values are:

0	absolute
1	scaled

6 VDC EXTENT: has 2 parameters:

P1: (point) first point
P2: (point) second point

If VDC TYPE is REAL, default VDC EXTENT is (0.0,0.0) , (0.9999..,0.9999..).

If VDC TYPE is INTEGER, default VDC EXTENT is (0,0) , (32767,32767).

7 BACKGROUND COLOUR: has 1 parameter:

P1: (direct colour) background colour, red,green,blue 3-tuple.

7.5 Control elements

Table 6 — Encoding of control elements.

Element Class 3	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
VDC INTEGER PRECISION	1	I	BI	16,24,32	16
VDC REAL PRECISION	2	E,2I	BE+2BI	{0,1}, {9,12,16,32}, {23,52,16,32}	1,16,16
AUXILIARY COLOUR	3	CO	BCO	COR	see below
TRANSPARENCY	4	E	BE	{0,1}	1
CLIP RECTANGLE	5	2P	2BP	VDCR	VDC EXTENT
CLIP INDICATOR	6	E	BE	{0,1}	1

Additional description of the elements in table 6:

Code Description

1 VDC INTEGER PRECISION: has 1 parameter:

P1: (integer) VDC integer precision; legal values are 16, 24 or 32; the value 8 is not permitted.

2 VDC REAL PRECISION: has 3 parameters:

P1: (enumerated) form of representation for real values: valid values are:

0 floating point format
1 fixed point format

P2: (integer) field width for exponent or whole part (including 1 bit for sign)
P3: (integer) field width for fraction or fractional part

Legal combinations of values are:

P1	P2	P3	Result
0	9	23	32-bit floating point
0	12	52	64-bit floating point
1	16	16	32-bit fixed point
1	32	32	64-bit fixed point

3 AUXILIARY COLOUR: has 1 parameter; its form depends on COLOUR SELECTION MODE:

P1: (colour) auxiliary colour. For direct colour selection, a 3-tuple of red, green and blue integer (I) values with range specified by the COLOUR PRECISION element; default is the device-dependent foreground colour. For indexed colour selection, an index (CIX), at COLOUR INDEX PRECISION, into the COLOUR TABLE; default is 0.

4 TRANSPARENCY: has 1 parameter:

P1: (enumerated) on-off indicator: valid values are:

0 off: auxiliary colour background is required

1	on: transparent background is required

5 CLIP RECTANGLE: has 2 parameters:

P1: (point) first point
P2: (point) second point

6 CLIP INDICATOR: has 1 parameter:

P1: (enumerated) clip indicator: valid values are:

0	off
1	on

7.6 Graphical primitive elements

Table 7 — Encoding of graphical primitive elements.

Element Class 4	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
POLYLINE	1	nP	nBP	VDCR	n/a
DISJOINT POLYLINE	2	nP	nBP	VDCR	n/a
POLYMARKER	3	nP	nBP	VDCR	n/a
TEXT	4	P,E,S	BP+BE+ BS	VDCR,{0,1}, SR	n/a
RESTRICTED TEXT	5	2VDC,P, E,S	2VDC,BP+ BE+BS	2VDCR,VDCR, {0,1},SR	n/a
APPEND TEXT	6	E,S	BE+BS	{0,1},SR	n/a
POLYGON	7	nP	nBP	VDCR	n/a
POLYGON SET	8	n(P,E)	n(BP+BE)	VDCR,0..3	n/a
CELL ARRAY	9	3P,3I, E,CLIST	3BP+3BI+ BE+nBCO	VDCR,+IR, {0,1},COR	n/a
GENERALIZED DRAWING PRIMITIVE	10	I,I,nP,D	2BI+nBP+ BS	IR,++IR, VDCR,SR	n/a
RECTANGLE	11	2P	2BP	VDCR	n/a
CIRCLE	12	P,VDC	BP+BVDC	VDCR, ++VDCR	n/a
CIRCULAR ARC 3 POINT	13	3P	3BP	VDCR	n/a
CIRCULAR ARC 3 POINT CLOSE	14	3P,E	3BP+BE	VDCR,{0,1}	n/a
CIRCULAR ARC CENTRE	15	P,4VDC, VDC	BP+4BVDC+ BVDC	VDCR,VDCR, ++VDCR	n/a
CIRCULAR ARC CENTRE CLOSE	16	P,4VDC, VDC,E	BP+4BVDC+ BVDC+BE	VDCR,VDCR, ++VDCR,{0,1}	n/a
ELLIPSE	17	3P	3BP	VDCR	n/a
ELLIPTICAL ARC	18	3P,4VDC	3BP+4BVDC	VDCR,VDCR	n/a
ELLIPTICAL ARC CLOSE	19	3P,4VDC, E	3BP+4BVDC+ BE	VDCR,VDCR, {0,1}	n/a

Additional description of the elements in table 7:

Code	Description
1	POLYLINE: has a variable parameter list:
	P1-Pn: (point) n (X,Y) polyline vertices
2	DISJOINT POLYLINE: has a variable parameter list:
	P1-Pn: (point) n (X,Y) line segment endpoints

3 POLYMARKER: has a variable parameter list:

P1-Pn: (point) n (X,Y) marker positions

4 TEXT: has 3 parameters:

P1: (point) text position
P2: (enumerated) final/not-final flag: valid values are:

0	not final
1	final

P3: (string) text string

5 RESTRICTED TEXT: has 5 parameters:

P1: (vdc) delta width
P2: (vdc) delta height
P3: (point) text position
P4: (enumerated) final/not-final flag: valid values are:

0	not final
1	final

P5: (string) text string

6 APPEND TEXT: has 2 parameters:

P1: (enumerated) final/not-final flag: valid values are:

0	not final
1	final

P2: (string) text string

7 POLYGON: has a variable parameter list:

P1-Pn: (point) n (X,Y) polygon vertices

8 POLYGON SET: has a variable parameter list of pairs of values, each of which has the following form:

P(i):(point) (X,Y) polygon vertex
P(i+1): (enumerated) edge out flag, indicating closures and edge visibility: valid values are

0	invisible
1	visible
2	close, invisible
3	close, visible

9 CELL ARRAY: has 8 parameters:

P1: (point) corner point P
P2: (point) corner point Q
P3: (point) corner point R
P4: (integer) nx
P5: (integer) ny
P6: (integer) local colour precision: valid values are 0, 1, 2, 4, 8, 16, 24, and 32. If the value is zero (the 'default colour precision indicator' value), the COLOUR (INDEX) PRECISION for the picture indicates the precision with which the colour list is encoded. If the value is non-zero, the precision with which the colour data is encoded is given by the value.
P7: (enumerated) cell representation mode: valid values are:

0 run length list mode

1 packed list mode

P8: (colour list) array of cell colour values.

If the COLOUR SELECTION MODE is 'direct', the values will be direct colour values. If the COLOUR SELECTION MODE is 'indexed', the values will be indexes into the COLOUR TABLE.

If the cell representation mode is 'packed list', the colour values are represented by rows of values, each row starting on a word boundary. If the cell representation mode is 'run length', the colour list values are represented by rows broken into runs of constant colour; each row starts on a word boundary. Each list item consists of a cell count (integer) followed by a colour value.

10 GENERALIZED DRAWING PRIMITIVE: has a variable parameter list:

P1: (integer) GDP identifier
P2: (integer) n, number of points in 'list of points'
P3-P(n+2): (point array) list of points
P(n+3)...: (data record) GDP data record

The parameter P2 is required to determine where the coordinate data ends and the data record begins. Data records are bound as strings in this encoding.

11 RECTANGLE: has 2 parameters:

P1: (point) first corner
P2: (point) second corner

12 CIRCLE: has 2 parameters:

P1: (point) centre of circle
P2: (vdc) radius of circle

13 CIRCULAR ARC 3 POINT: has 3 parameters:

P1: (point) starting point
P2: (point) intermediate point
P3: (point) ending point

14 CIRCULAR ARC 3 POINT CLOSE: has 4 parameters:

P1: (point) starting point
P2: (point) intermediate point
P3: (point) ending point
P4: (enumerated) type of arc closure: valid values are:

0 pie closure
1 chord closure

15 CIRCULAR ARC CENTRE: has 6 parameters:

P1: (point) centre of circle
P2: (vdc) delta X for start vector
P3: (vdc) delta Y for start vector
P4: (vdc) delta X for end vector
P5: (vdc) delta Y for end vector
P6: (vdc) radius of circle

16 CIRCULAR ARC CENTRE CLOSE: has 7 parameters:

P1: (point) centre of circle
P2: (vdc) delta X for start vector
P3: (vdc) delta Y for start vector
P4: (vdc) delta X for end vector

P5: (vdc) delta Y for end vector
P6: (vdc) radius of circle
P7: (enumerated) type of arc closure: valid values are:

0	pie closure
1	chord closure

17 ELLIPSE: has 3 parameters:

P1: (point) centre of ellipse
P2: (point) endpoint of first conjugate diameter
P3: (point) endpoint of second conjugate diameter

18 ELLIPTICAL ARC: has 7 parameters:

P1: (point) centre of ellipse
P2: (point) endpoint for first conjugate diameter
P3: (point) endpoint for second conjugate diameter
P4: (vdc) delta X for start vector
P5: (vdc) delta Y for start vector
P6: (vdc) delta X for end vector
P7: (vdc) delta Y for end vector

19 ELLIPTICAL ARC CLOSE: has 8 parameters:

P1: (point) centre of ellipse
P2: (point) endpoint for first conjugate diameter
P3: (point) endpoint for second conjugate diameter
P4: (vdc) delta X for start vector
P5: (vdc) delta Y for start vector
P6: (vdc) delta X for end vector
P7: (vdc) delta Y for end vector
P8: (enumerated) type of arc closure: valid values are:

0	pie closure
1	chord closure

7.7 Attribute elements

Table 8 — Encoding of attribute elements.

Element Class 5	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
LINE BUNDLE INDEX	1	IX	BIX	+IXR	1
LINE TYPE	2	IX	BIX	IXR	1
LINE WIDTH	3	VDC or R	BVDC or BR	++VDCR or ++RR	see below
LINE COLOUR	4	CO	BCO	COR	see below
MARKER BUNDLE INDEX	5	IX	BIX	+IXR	1
MARKER TYPE	6	IX	BIX	IXR	3
MARKER SIZE	7	VDC or R	BVDC or BR	++VDCR or ++RR	see below
MARKER COLOUR	8	CO	BCO	COR	see below
TEXT BUNDLE INDEX	9	IX	BIX	+IXR	1
TEXT FONT INDEX	10	IX	BIX	+IXR	1
TEXT PRECISION	11	E	BE	0..2	0
CHARACTER EXPANSION FACTOR	12	R	BR	+RR	1.0
CHARACTER SPACING	13	R	BR	RR	0.0
TEXT COLOUR	14	CO	BCO	COR	see below
CHARACTER HEIGHT	15	VDC	BVDC	++VDCR	see below
CHARACTER ORIENTATION	16	4VDC	4BVDC	VDCR	0,1,1,0
TEXT PATH	17	E	BE	0..3	0
TEXT ALIGNMENT	18	2E, R,R	2BE+ 2BR	0..4, 0..6, 2RR	0,0, 0.0,0.0
CHARACTER SET INDEX	19	IX	BIX	+IXR	1
ALTERNATE CHARACTER SET INDEX	20	IX	BIX	+IXR	1

(continued)

Table 8 — Encoding of attribute elements (concluded).

FILL BUNDLE INDEX	21	IX	BIX	+IXR	1
INTERIOR STYLE	22	E	BE	ER	0
FILL COLOUR	23	CO	BCO	COR	see below
HATCH INDEX	24	IX	BIX	IXR	1
PATTERN INDEX	25	IX	BIX	+IXR	1
EDGE BUNDLE INDEX	26	IX	BIX	+IXR	1
EDGE TYPE	27	IX	BIX	IXR	1
EDGE WIDTH	28	VDC or R	BVDC or BR	++VDCR or ++RR	see below
EDGE COLOUR	29	CO	BCO	COR	see below
EDGE VISIBILITY	30	E	BE	{0,1}	0
FILL REFERENCE POINT	31	P	BP	VDCR	0,0
PATTERN TABLE	32	IX,3I, nx*nyCO	BIX+3BI+ nx*nyBCO	+IXR,+IR COR	1,(1,1), 0,0
PATTERN SIZE	33	4VDC	4BVDC	VDCR	see below
COLOUR TABLE	34	CI,nCD	BCI+nBCD	CIR,CDR	see below
ASPECT SOURCE FLAGS	35	n(E,E)	n(2BE)	(0..17), {0,1}	(n,0), n=0..17

Additional description of the elements in table 8:

Code Description

1 LINE BUNDLE INDEX: has 1 parameter:

P1: (index) line bundle index

2 LINE TYPE: has 1 parameter:

P1: (index) line type: the following values are standardized:

1	solid
2	dash
3	dot
4	dash-dot
5	dash-dot-dot

negative for private use

3 LINE WIDTH: has 1 parameter:

P1: (vdc) line width (if size specification is 'absolute') or (real) line width scale factor (if size specification is 'scaled'). Default is 1.0 for 'scaled', or for 'absolute' 0.001 times the longest side of the default VDC EXTENT.

4 LINE COLOUR: has 1 parameter; its form depends on COLOUR SELECTION MODE:

P1: (colour) line colour. For direct colour selection, a 3-tuple of red, green and blue integer values (CD) with range specified by the COLOUR PRECISION element; default is the device-dependent foreground colour. For indexed colour selection, an index (CIX), at COLOUR INDEX PRECISION, into the COLOUR TABLE; default is 1.

5 MARKER BUNDLE INDEX: has 1 parameter:

P1: (index) marker bundle index

6 MARKER TYPE: has 1 parameter:

P1: (index) marker type: the following values are standardized:

1	dot
2	plus
3	asterisk
4	circle
5	cross

negative for private use

7 MARKER SIZE: has 1 parameter:

P1: (vdc) marker size (if size specification is 'absolute') or (real) marker size scale factor (if size specification is 'scaled'). Default is 1.0 for 'scaled', or for 'absolute' 0.01 times the longest side of the default VDC EXTENT.

8 MARKER COLOUR: has 1 parameter; its form depends on COLOUR SELECTION MODE:

P1: (colour) marker colour. For direct colour selection, a 3-tuple of red, green and blue integer values (CD) with range specified by the COLOUR PRECISION element; default is the device-dependent foreground colour. For indexed colour selection, an index (CIX), at COLOUR INDEX PRECISION, into the COLOUR TABLE; default is 1.

9 TEXT BUNDLE INDEX: has 1 parameter:

P1: (index) text bundle index

10 TEXT FONT INDEX: has 1 parameter:

P1: (index) text font index

11 TEXT PRECISION: has 1 parameter:

P1: (enumerated) text precision: valid values are:

0	string
1	character
2	stroke

12 CHARACTER EXPANSION FACTOR: has 1 parameter:

P1: (real) character expansion factor

13 CHARACTER SPACING: has 1 parameter:

P1: (real) additional inter-character space

14 TEXT COLOUR: has 1 parameter; its form depends on COLOUR SELECTION MODE:

P1: (colour) text colour. For direct colour selection, a 3-tuple of red, green and blue integer values (CD) with range specified by the COLOUR PRECISION element; default is the device-dependent foreground colour. For indexed colour selection, an index (CIX), at COLOUR INDEX PRECISION, into the COLOUR TABLE; default is 1.

15 CHARACTER HEIGHT: has 1 parameter:

P1: (vdc) character height. Default is 0.01 times the longest side of the default VDC EXTENT.

16 CHARACTER ORIENTATION: has 4 parameters:

P1: (vdc) X character up component
P2: (vdc) Y character up component
P3: (vdc) X character base component
P4: (vdc) Y character base component

17 TEXT PATH: has 1 parameter:

P1: (enumerated) text path: valid values are:

0	right
1	left
2	up
3	down

18 TEXT ALIGNMENT: has 4 parameters:

P1: (enumerated) horizontal alignment: valid values ares:

0	normal
1	left
2	centre
3	right
4	continuous horizontal

P2: (enumerated) vertical alignment

0	normal
1	top
2	cap
3	half
4	base
5	bottom
6	continuous vertical

P3: (real) continuous horizontal alignment
P4: (real) continuous vertical alignment

19 CHARACTER SET INDEX: has 1 parameter:

P1: (index) character set index

20 ALTERNATE CHARACTER SET INDEX: has 1 parameter:

P1: (index) alternate character set index

21 FILL BUNDLE INDEX: has 1 parameter:

P1: (index) fill bundle index

22 INTERIOR STYLE: has 1 parameter:

P1: (enumerated) interior style: valid values are:

0	hollow
1	solid
2	pattern
3	hatch
4	empty

negative for private use

23 FILL COLOUR: has 1 parameter; its form depends on COLOUR SELECTION MODE:

P1: (colour) fill colour. For direct colour selection, a 3-tuple of red, green and blue integer values (CD) with range specified by the COLOUR PRECISION element; default is the device-dependent foreground colour. For indexed colour selection, an index (CIX), at COLOUR INDEX PRECISION, into the COLOUR TABLE; default is 1.

24 HATCH INDEX: has 1 parameter

P1: (index) hatch index: the following values are standardized:

1	horizontal
2	vertical

3 positive slope
4 negative slope
5 combined vertical and horizontal slant
6 combined left and right slant
negative for private use

25 PATTERN INDEX: has 1 parameter

P1: (index) pattern index

26 EDGE BUNDLE INDEX: has 1 parameter:

P1: (index) edge bundle index

27 EDGE TYPE: has 1 parameter:

P1: (integer) edge type: the following values are standardized:

1 solid
2 dash
3 dot
4 dash-dot
5 dash-dot-dot
negative for private use

28 EDGE WIDTH: has 1 parameter:

P1: (vdc) edge width (if size specification is absolute) or (real) edge width scale factor (if size specification is scaled). Default is 1.0 for 'scaled', or for 'absolute' 0.001 times the longest side of the default VDC EXTENT.

29 EDGE COLOUR: has 1 parameter; its form depends on COLOUR SELECTION MODE:

P1: (colour) edge colour. For direct colour selection, a 3-tuple of red, green and blue integer values (CD) with range specified by the COLOUR PRECISION element; default is the device-dependent foreground colour. For indexed colour selection, an index (CIX), at COLOUR INDEX PRECISION, into the COLOUR TABLE; default is 1.

30 EDGE VISIBILITY: has 1 parameter:

P1: (enumerated) edge visibility: valid values are:

0 off
1 on

31 FILL REFERENCE POINT: has 1 parameters:

P1: (point) fill reference point

32 PATTERN TABLE: has 5 parameters:

P1: (index) pattern table index
P2: (integer) nx, the dimension of colour array in the direction of the PATTERN SIZE width vector
P3: (integer) ny, the dimension of colour array in the direction of the PATTERN SIZE height vector
P4: (integer) local colour precision: valid values are 0, 1, 2, 4, 8, 16, 24, and 32. If the value is zero (the 'default colour precision indicator' value), the COLOUR (INDEX) PRECISION for the picture indicates the precision with which the colour list is encoded. If the value is non-zero, the precision with which the colour data is encoded is given by the value.
P5: (colour array) pattern definition

33 PATTERN SIZE: has 4 parameters:

P1: (vdc) pattern height vector, x component
P2: (vdc) pattern height vector, y component

P3: (vdc) pattern width vector, x component
P4: (vdc) pattern width vector, y component

The default pattern size is 0,dy,dx,0, where dx and dy are respectively the height and width of the VDC extent.

34 COLOUR TABLE: has 2 parameters:

P1: (colour index) starting colour table index
P2: (direct colour list) list of direct colour values (red,green,blue 3-tuples)

35 ASPECT SOURCE FLAGS: has up to 18 parameter-pairs, corresponding to each attribute that may be bundled; each parameter-pair contains the ASF type and the ASF value:

(enumerated) ASF type; valid values are:

0	line type ASF
1	line width ASF
2	line colour ASF
3	marker type ASF
4	marker size ASF
5	marker colour ASF
6	text font index ASF
7	text precision ASF
8	character expansion factor ASF
9	character spacing ASF
10	text colour ASF
11	interior style ASF
12	fill colour ASF
13	hatch index ASF
14	pattern index ASF
15	edge type ASF
16	edge width ASF
17	edge colour ASF

(enumerated) ASF value; valid values are:

0	individual
1	bundled

7.8 Escape element

Table 9 — Encoding of escape element.

Element Class 6	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
ESCAPE	1	I,D	BI+BS	IR,SR	n/a

Additional description of the elements in table 9:

Code	Description
1	ESCAPE: has 2 parameters: P1: (integer) escape identifier P2: (data record) escape data record; data records are bound as strings in this encoding.

7.9 External elements

Table 10 — Encoding of external elements.

Element Class 7	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
MESSAGE	1	E,S	BE+BS	{0,1},SR	0 or n/a
APPLICATION DATA	2	I,D	BI+BS	IR,SR	n/a

Additional description of the elements in table 10:

Code	Description
1	MESSAGE: has 2 parameters:

P1: (enumerated) action-required flag: valid values are:

0	no action
1	action

P2: (string) message string

2 APPLICATION DATA: has 2 parameters:

P1: (integer) identifier
P2: (data record) application data record; data records are bound as strings in this encoding.