INTERNATIONAL STANDARD

ISO/IEC 8632-3

Second edition
1999-12-15

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

Part 3: Binary encoding

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

Partie 3: Codage binaire

Reference number
ISO/IEC 8632-3:1999(E)

PDF disclaimer

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

Adobe is a trademark of Adobe Systems Incorporated.

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

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

Printed in Switzerland

Contents

Foreword

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

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

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

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 8632 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 8632-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 24, *Computer graphics and image processing*.

This second edition cancels and replaces the first edition (ISO/IEC 8632-3:1992), which has been technically revised. It also incorporates Amendment 1:1994 and Amendment 2:1995. Note that the previous edition of ISO/IEC 8632-3, published in 1992, was a first edition but second edition was indicated by error on its cover page and in the foreword.

ISO/IEC 8632 consists of the following parts, under the general title *Information technology — Computer graphics — Metafile for the storage and transfer of picture description information*:

— *Part 1: Functional specification*

— *Part 3: Binary encoding*

— *Part 4: Clear text encoding*

Annex A forms a normative part of this part of ISO/IEC 8632. Annexes B and C are for information only.

NOTE In previous editions of ISO/IEC 8632, Part 2 defined a Character Encoding. Part 2 was withdrawn in 1998, due to its lack of implementation and use.

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 another encoding.

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/IEC 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 5); 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. When the data of an element (whether partitioned or not) does not terminate on an even-octet boundary, then the following element is aligned by padding after the data of the preceding element with zero bits to the next even-octet boundary. A no-op element is available in this encoding. It is skipped and ignored by interpreters. 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 minimises 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 words 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 a cell count is specified followed by the colour (or colour index).

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/IEC 8632 is that in ANSI/IEEE 754-1986.

The representation of character data in this part of ISO/IEC 8632 follows the rules of ISO/IEC 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/IEC 8632-1, 6.12.)

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

Part 3:
Binary encoding

1 Scope

This part of ISO/IEC 8632 specifies a binary encoding of the Computer Graphics Metafile. For each of the elements specified in ISO/IEC 8632-1, this part specifies an encoding in terms of data types.

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 Conformance

Conformance of metafiles to ISO/IEC 8632 is defined in terms of profiles. A metafile conforms to this encoding if it conforms to a profile and meets the following criteria:

— Each metafile element described in this part shall be encoded in the manner described in this part of this International Standard and a profile.

— Metafile elements which are not defined in Part 1 or in this encoding are all encoded using the GENERALIZED DRAWING PRIMITIVE or ESCAPE metafile elements as appropriate. According to the profile rules of Part 1 (see clause 9, subclause 9.5.2.8), such elements shall either be profile defined or registered, in order that the profile be valid. Inclusion of private elements is not permissible in a valid profile of ISO/IEC 8632 and this encoding.

— Values of index parameters, which are used as enumeration selectors from lists of implicitly defined attribute values, shall either be standard, registered, or profile defined. The standard and registered values are all non-negative, and the profile-defined shall be negative. Use of private, implicitly-defined negative index values which are not profile defined is not permissible in a valid profile of ISO/IEC 8632 and this encoding.

— Values specified as being "reserved for registered values" shall not be used unless their meaning has been registered or standardized.

— Inclusion of non-graphical data in the metafile shall be accomplished with the APPLICATION DATA element or with the APPLICATION STRUCTURE ATTRIBUTE element.

See clause 10 for additional conformance information about this encoding.

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 8632. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 8632 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 646:1991, *Information technology — ISO 7-bit coded character set for information interchange.*

ISO 2022:1986, *Information processing — ISO 7-bit and 8-bit coded character sets — Code extension techniques.*

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

4 Notational conventions

“Command Header” is used throughout this part of ISO/IEC 8632 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 5).

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).

5 Overall structure

5.1 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.

5.2 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), 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

5.3 General structure of the binary metafile

The binary encoding of the metafile is a logical data structure consisting of a sequential collection of bits.

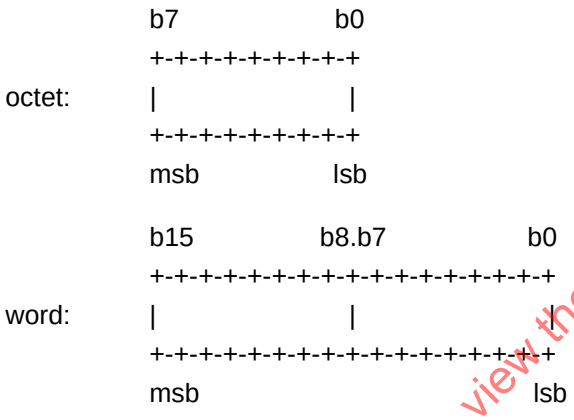
For convenience in describing the length and alignment of metafile elements, fields of two different sizes are defined within the structure. These fields are used in the remainder of this part of ISO/IEC 8632 for illustrating the contents and structure of elements and parameters.

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.



In the preceding diagram, msb means most significant bit and lsb means least significant bit.

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
.	.	.
.	.	.

5.4 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.

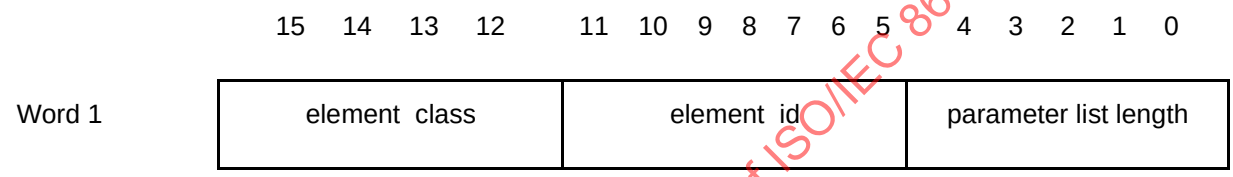


Figure 1 — Format of a short-form Command Header

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 the 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.

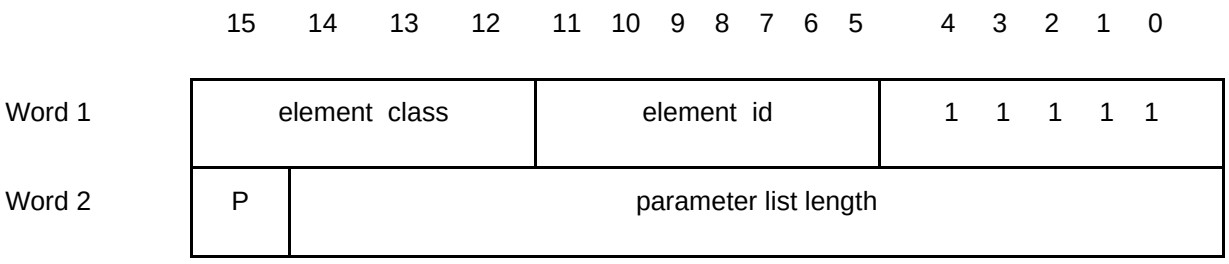


Figure 2 — Format of a long-form Command Header

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 of ISO/IEC 8632. 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 ISO/IEC 8632-1. 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 ISO/IEC 8632-1.

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 this part of ISO/IEC 8632. 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 this part of ISO/IEC 8632.

6 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/IEC 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 6.1 to 6.5 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

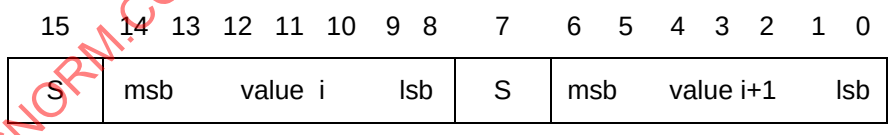
The data types in the following data diagrams are illustrated for the case that the parameter begins on a metafile word boundary. In general, parameters may align on odd or even octet boundaries, because they may be preceded by an odd or even number of octets of other parameter data. Elements containing 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.

6.1 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.) In the diagrams of the following subsections, 'value’ indicates the value for positive integers and the two’s complement of the value for negative integers.

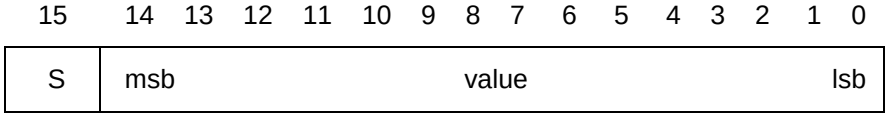
6.1.1 Signed integer at 8-bit precision

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



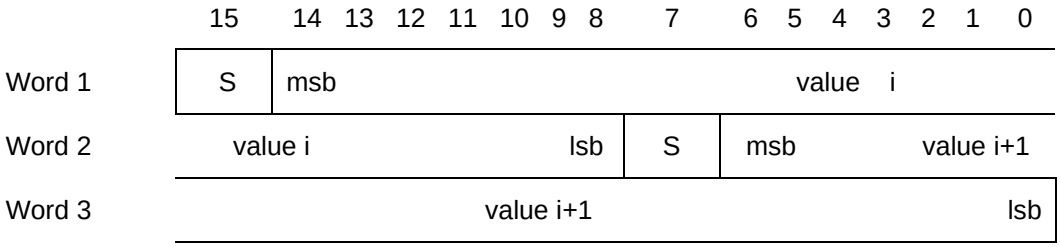
6.1.2 Signed integer at 16-bit precision

Each value occupies one metafile word.



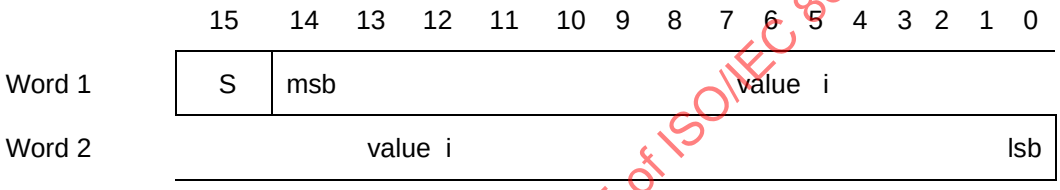
6.1.3 Signed integer at 24-bit precision

Each value straddles two successive metafile words.



6.1.4 Signed integer at 32-bit precision

Each value fills two complete metafile words.

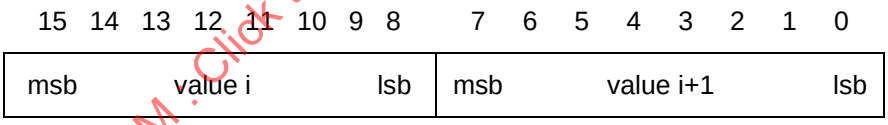


6.2 Unsigned integer

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

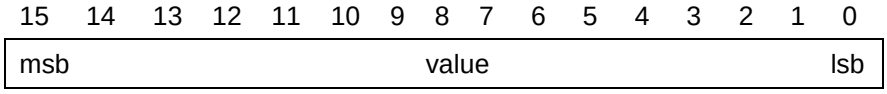
6.2.1 Unsigned integers at 8-bit precision

Each value occupies half a metafile word.



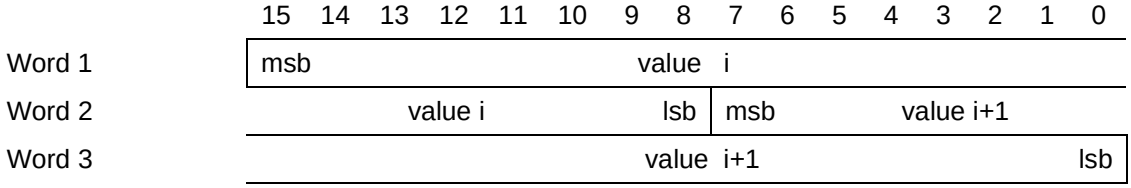
6.2.2 Unsigned integers at 16-bit precision

Each value occupies one metafile word.



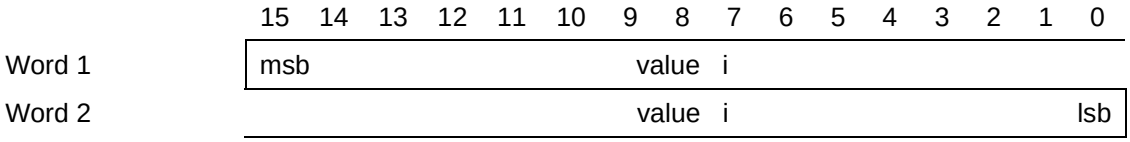
6.2.3 Unsigned integers at 24-bit precision

Each value straddles two successive metafile words.



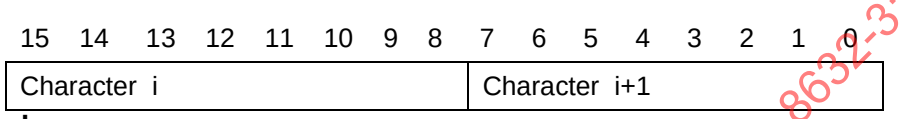
6.2.4 Unsigned integers at 32-bit precision

Each value fills two complete metafile words.



6.3 Character

Each character is stored in 1 or more consecutive octets, depending upon the coding of the particular character set. The following illustrates characters which are coded with 1 octet each.

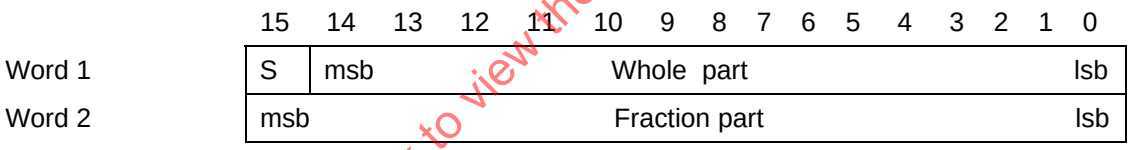


6.4 Fixed point real

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 6.1); the second represents the “fractional part” and has the same form as an Unsigned Integer (UI; see 6.2). Two precisions may be specified for Fixed Point Reals: 32-bit or 64-bit.

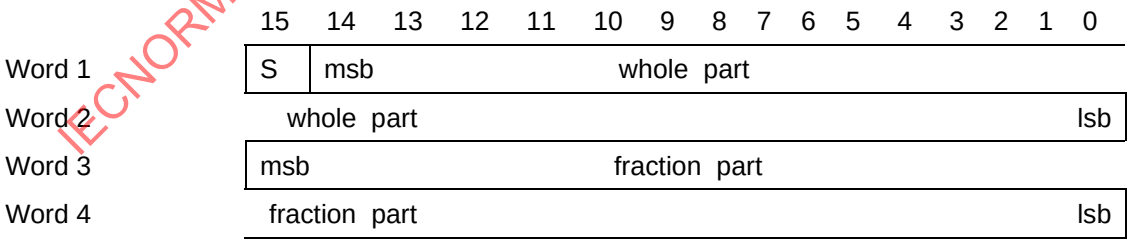
6.4.1 Fixed point real at 32-bit precision

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.



6.4.2 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.



6.4.3 Value of fixed point reals

The values of the represented real numbers are given by:

for 32 bits: real_value = $SI + \left[\frac{UI}{2^{16}}\right]$

for 64 bits: 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.

6.5 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 ≠ 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 ≠ 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 ≠ 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 ≠ 0, then the magnitude of the value is $(0.f)(2^{-1022})$.

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

6.5.1 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

6.5.2 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

7 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/IEC 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/IEC 8632-1.
3) The symbol for the number of octets required to represent one instance (occurrence) 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 negative integers, range of negative 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.

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)}
CCO	UI at direct colour precision (dcp)	BCCO {=dcp/8}	CCOR {0..(2^{dcp} - 1)} {see NOTE 2}
CD	(CCO,CCO,CCO) or (CCO,CCO,CCO,CCO)	BCD ={3*BCCO} BCD ={4*BCCO}	CCOR {see NOTE 1, NOTE 16}
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}	{-2^{15} to 2^{15} - 1 } {see NOTE 18}
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 NOTE 5, NOTE 10}
S,SF,D	UI,nC	BS {see NOTE }	SR {see NOTE 6, NOTE 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 NOTE 1, NOTE 5, NOTE 7, NOTE 8}
P	(VDC,VDC)	BP {=2*BVDC}	VDCR {see NOTE 1, NOTE 5, NOTE 7, NOTE 8}
CO	CI or CD	BCO {=BCI} or BCO {=BCD}	COR {=CIR} {see NOTE 9, NOTE 11} or COR {=CCOR}
N	SI at name precision (np)	BN {=np/8}	NR {-2^{np-1} to 2^{np-1} - 1}
VC	I or R	BVC {=BI} or BVC {=BR}	VCR {=IR} {see NOTE 13} or VCR {=RR}
VP	(VC,VC)	BVP {=2*BVC}	VCR {see NOTE 1, NOTE 13, NOTE 14}

Table 1 — Representation of abstract data types (continued)

Abstract symbol	Parameter construction from primitive forms	Octets per parameter: symbol and value	Parameter range: symbol and value
BS	nUI at fixed precision (16-bit) {see NOTE 15}	BBS {=2n}	BSR {see NOTE 15}
UI8	UI at fixed precision (8-bit)	BUI8 {=1}	UI8R {0..255}
UI32	UI at fixed precision (32-bit)	BUI32 {=4}	UI32R {0..2^{32} - 1}
SDR	{see NOTE 17}	BSDR	n/a
SS	VDC or R	BSS {=BVDC} or BSS {=BR}	SSR {=VDCR} {see NOTE 19} or SSR {=RR}

The following 19 notes contain additional normative specifications of this encoding:

NOTE 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.

NOTE 2 For colour models RGB and CMYK a direct colour component is abstractly a real in the range [0,1]. For colour models CIELAB, CIELUV, and RGB-related it is abstractly a real in the respective colour spaces with possibly different ranges for the direct colour components. The COLOUR VALUE EXTENT element provides for the mapping between a direct colour component represented as UI and the corresponding real value.

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

NOTE 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.

NOTE 5 FPR and VDCR (when VDC are floating point reals) are computed following the ANSI/IEEE 754 floating point standard (see clause 6 on the floating point data form).

NOTE 6 The range for parameter types S and SF is not applicable. The range for character data is not applicable. A string is encoded as a count (unsigned integer) followed by characters. The count is a count of octets in the string, not whole character codes (the two are equal for single byte codes, but not for multi-byte codes).

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.

If the number of whole character codes in a string is n, and the number of octets per character code is constant within the string and equal to m, and if the string is not continued (as a long-form string may be), then the number of octets in the string parameter is either n·m+1 or n·m+3, depending upon whether the string is short-form or long-form, respectively. If the number of octets per character code is not constant and/or the string is a continued long-form string, then the number of octets is the string is not so easily expressed, but is the total of the octets used in the "data" part of the string and the number of octets used for length information.

Table 1 — Representation of abstract data types (continued)

NOTE 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.

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

NOTE 9 The parameter type symbol CO corresponds to the data type CO of ISO/IEC 8632-1. It 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.

NOTE 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 all metafile parameters of type R except for:

a) the *metric scale factor* parameter of the SCALING MODE element;

b) the *metric scale factor* parameter of the DEVICE VIEWPORT SPECIFICATION MODE element;

In this encoding, these parameters are always encoded as floating point.

NOTE 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.

For all rows of the cell array, except possibly the last row, the colour data thus occupies $2n_y$ (1+ [(p・n_x - 1) / 16]) octets, where nx is the number of cells per row, ny is the number of rows, p is the number of bits per colour, and [..] denotes "the greatest integer in .." Because the last row does not have a subsequent row which must align on a word boundary, which alignment (for all other rows but the last) potentially requires the addition to the end of the row of a padding byte, the color data of the last row occupies n_y (1+ [(p・n_x - 1) / 8]) octets (however, see clause 10).

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>*. With the exception of the first run of a row, the integer count of each run immediately follows the colour specifier of the preceding run with no intervening padding.

NOTE 12 Abstract parameter type Data Record, D, is encoded in this part similarly to string data. However, the constraints on character code values and the character set switching mechanisms (both those related to CHARACTER SET INDEX, and the purely ISO 2022 switching methods) do not necessarily apply to data records, as they do to the structurally similar S and SF parameters.

How the data are encoded, the meaning of the data bytes in the record, and the effect (if any) of character set switching mechanisms are part of the definitions of the individual Escape, GDP, and External elements to which the data record belongs.

The coding technique of the SDR data type (see Table 1, NOTE 17) is one valid form for a Data Record parameter. This form is recommended for GDP, Escape, and External element proposals submitted for Graphical Registration.

The coding tables in clause 8 will use the symbol D for the parameter type, and will use the S-related symbols for other information about the parameter.

Table 1 — Representation of abstract data types (continued)

NOTE 13 The abstract parameter type VC, a single VC value, is either a real or an integer, depending on the declaration of the picture descriptor element DEVICE VIEWPORT SPECIFICATION MODE. When DEVICE VIEWPORT SPECIFICATION MODE is 'fraction of display surface', the value is real. When DEVICE VIEWPORT SPECIFICATION MODE is 'millimetres with scale factor' or 'physical device coordinates', the value is integer. Subsequent tables use a single set of values, VC, BVC and VCR, recognizing that they are computed differently depending on DEVICE VIEWPORT SPECIFICATION MODE.

NOTE 14 The abstract parameter type VC is a single value; a viewport point, VP, is an ordered pair of VC.

NOTE 15 The bitstream (BS) data type is encoded as a stream of binary digits (bits) packed in 16-bit unsigned integers. The BS data type is used in part 1 of this Standard for the compressed colour specifier lists of Tile Array elements. A bitstream type parameter shall be encoded in the Binary Encoding of this part with the smallest number of whole 16-bit words which will hold the bits of the parameter data. If the parameter data bits do not exactly fill an integral number of 16-bit words, the remaining bits in the last word shall be 0. The range for parameter type BS is not applicable.

NOTE 16 The abstract parameter type CD is a 3-tuple or 4-tuple of CCO depending on COLOUR MODEL.

NOTE 17 The structured data record (SDR) of part 1 of this International Standard is composed entirely of other standardized datatypes (including SDR itself) in a structure which is self-defining. SDR is encoded by encoding each of the component operands according to the normal encoding rules for its corresponding data type. The string of octets comprising the encoded operands is then treated as an operand of type S — it is preceded by a string count, short form or long form, and can be continued if long form (see NOTE 6, above).

NOTE 18 Ranges for enumerated and index parameters indicated by {n..m} in tables 3-10 refer to standardized values. Index ranges are subject to extension by registration.

NOTE 19 The parameter type symbol SS corresponds to parameter type SS of ISO/IEC 8632-1. It is not a basic data type. It is a shorthand for data which can be VDC or Real, depending upon an associated specification mode. The associations of these modes with the various element parameters are defined in subclause 7.1 of part 1.

8 Representation of each element

8.1 Method of presentation

The elements are grouped according to their class; there are 10 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	Segment Control and Segment Attribute elements
9	Application Structure Descriptor elements
10-15	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 subclause which consists of a table and a set of notes. The table specifies the metafile element, element id, parameter type, parameter list length, and parameter range. The parameter list length is given in octets, which in some cases is constant and in other cases is variable. Any element that does not consist of an even number octets is padded with zero bits to the next 16-bit boundary before the command header of the next element is written to the metafile — elements begin on 16-bit boundaries.

The defaults for the elements are as given in clause 8 of ISO/IEC 8632-1.

This clause specifies some of the constraints on parameter values. The specifications are not exhaustive, for example such constraints as the non-collinearity of text vectors are not stated. All parameter value and other element state constraints of ISO/IEC 8632-1, including those of the formal grammars, shall apply to metafiles encoded according to this part.

8.2 Delimiter elements

Table 3 — Encoding of delimiter elements

Element Class 0	Element Id	Parameter Type	Parameter List Length	Parameter Range
no-op	0	see below	n	n/a
BEGIN METAFILE	1	SF	BS	SR
END METAFILE	2	n/a	0	n/a
BEGIN PICTURE	3	SF	BS	SR
BEGIN PICTURE BODY	4	n/a	0	n/a
END PICTURE	5	n/a	0	n/a
BEGIN SEGMENT	6	N	BN	NR
END SEGMENT	7	n/a	0	n/a
BEGIN FIGURE	8	n/a	0	n/a
END FIGURE	9	n/a	0	n/a
BEGIN PROTECTION REGION	13	IX	BIX	+IXR
END PROTECTION REGION	14	n/a	0	n/a
BEGIN COMPOUND LINE	15	n/a	0	n/a
END COMPOUND LINE	16	n/a	0	n/a
BEGIN COMPOUND TEXT PATH	17	n/a	0	n/a
END COMPOUND TEXT PATH	18	n/a	0	n/a
BEGIN TILE ARRAY	19	P, 2E, 4I,2R, 2I,2I	BP+ 2BE+ 4BI+2BR+ 4BI	VDCR, {0..3},{0,1} +IR,+RR ++IR,+IR
END TILE ARRAY	20	n/a	0	n/a
BEGIN APPLICATION STRUCTURE	21	SF,SF,E	2BS+BE	SR,SR,{0,1}
BEGIN APPLICATION STRUCTURE BODY	22	n/a	0	n/a
END APPLICATION STRUCTURE	23	n/a	0	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 fixed) metafile name
2	END METAFILE: has no parameters.
3	BEGIN PICTURE: has 1 parameter: P1: (string fixed) picture name
4	BEGIN PICTURE BODY: has no parameters.
5	END PICTURE: has no parameters.
6	BEGIN SEGMENT: has 1 parameter: P1: (name) segment identifier
7	END SEGMENT: has no parameters
8	BEGIN FIGURE: has no parameters
9	END FIGURE: has no parameters
13	BEGIN PROTECTION REGION: has 1 parameter: P1: (index) region index.

14 END PROTECTION REGION: has no parameters.

15 BEGIN COMPOUND LINE: has no parameters:

16 END COMPOUND LINE: has no parameters.

17 BEGIN COMPOUND TEXT PATH: has no parameters:

18 END COMPOUND TEXT PATH: has no parameters.

19 BEGIN TILE ARRAY: has 13 parameters:
P1: (point) position.
P2: (enumerated) cell path direction: valid values are

0	0°
1	90°
2	180°
3	270°

P3: (enumerated) line progression direction: valid values are

0	90°
1	270°

P4: (integer) number of tiles in pth direction.
P5: (integer) number of tiles in line direction.
P6: (integer) number of cells/tile in path direction.
P7: (integer) number of cells/tile in line direction.
P8: (real) cell size in path direction.
P9: (real) cell size in line direction.
P10: (integer) image offset in path direction.
P11: (integer) image offset in line direction.
P12: (integer) image number of cells in path direction.
P13: (integer) image number of cells in line direction.

20 END TILE ARRAY: has no parameters.

21 BEGIN APPLICATION STRUCTURE:has three parameters
P1: (string fixed)application structure identifier
P2: (string fixed)application structure type
P3: (enumerated)inheritance flag:valid values are

0	STATE LIST
1	APPLICATION STRUCTURE

22 BEGIN APPLICATION STRUCTURE BODY:has no parameters

23 END APPLICATION STRUCTURE:has no parameters

8.3 Metafile descriptor elements

Table 4 — Encoding of metafile descriptor elements

Element Class 1	Element Id	Parameter Type	Parameter List Length	Parameter Range
METAFILE VERSION	1	I	BI	+IR(1..n)
METAFILE DESCRIPTION	2	SF	BS	SR
VDC TYPE	3	E	BE	0,1
INTEGER PRECISION	4	I	BI	8,16,24,32
REAL PRECISION	5	E,2I	BE+2BI	{0,1}, {9,12,16,32}, {23,52,16,32}
INDEX PRECISION	6	I	BI	8,16,24,32
COLOUR PRECISION	7	I	BI	8,16,24,32
COLOUR INDEX PRECISION	8	I	BI	8,16,24,32
MAXIMUM COLOUR INDEX	9	CI	BCI	CIR
COLOUR VALUE EXTENT	10	2CD or 6R	2BCD or 6BR	CCOR or RR
METAFILE ELEMENT LIST	11	I,2nIX	BI,2nBIX	++IR,IXR
METAFILE DEFAULTS REPLACEMENT	12	Metafile elements	variable	Metafile elements
FONT LIST	13	nSF	nBS	SR
CHARACTER SET LIST	14	n(E,SF)	n(BE+BS)	{0..4},SR
CHARACTER CODING ANNOUNCER	15	E	BE	0,1,2,3
NAME PRECISION	16	I	BI	8,16,24,32
MAXIMUM VDC EXTENT	17	2P	2BP	VDCR
SEGMENT PRIORITY EXTENT	18	2I	2BI	++IR
COLOUR MODEL	19	IX	BIX	+IXR
COLOUR CALIBRATION	20	IX,3R, 18R,I, 6nCCO,I, mCD,3mR	BIX+3BR+ 18BR+BI+ 6nBCCO+BI+ mBCD+3mBR	+IXR,RR, RR,++IR, CCOR,++IR, CCOR,RR
FONT PROPERTIES	21	n[IX,I,SDR]	n(BIX+BI)+ (sum of)BSDR	n/a
GLYPH MAPPING	22	IX,E, SF,I, IX,SDR	BIX+BE+ BS+BI+ BIX+BSDR	+IXR,ER SR,+IR IXR,n/a
SYMBOL LIBRARY LIST	23	nSF	nBS	SR
PICTURE DIRECTORY	24	E,n(SF,2[Idt])	BE+n(BS+2B[Idt])	{0,1,2}, (SR,[Idt]R,[Idt]R)

Additional description of the elements in table 4:

Code Description

1 METAFILE VERSION: has 1 parameter:

P1: (integer) metafile version number: valid values are 1, 2, 3, 4

2 METAFILE DESCRIPTION: has 1 parameter:

P1: (string fixed) metafile description 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: valid values are 8, 16, 24 or 32

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 variable parameters depending upon the colour model:

If the model is RGB or CMYK, then 2 parameters:

P1: (direct colour value) minimum colour value

P2: (direct colour value) maximum colour value

If the model is CIELAB, CIELUV, or RGB-related then 3 parameters:

P1: (real) scale and offset pair for first component.

P2: (real) scale and offset pair for second component.

P3: (real) scale and offset pair for third component.

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 Table 3 to Table 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)
version-2 set:	(-1,2)
extended-primitives set:	(-1,3)
version-2-gksm set:	(-1,4)
version-3 set:	(-1,5)
version-4 set	(-1,6)

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: (string fixed) n font names

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

P2: (string fixed) Designation sequence tail; see Part 1, subclause 7.3.14.

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

16 NAME PRECISION: has 1 parameter:

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

17 MAXIMUM VDC EXTENT: has 2 parameters:

P1: (point) first corner

P2: (point) second corner

18 SEGMENT PRIORITY EXTENT: has 2 parameters:

P1: (integer) minimum segment priority value: valid values are non-negative integers

P2: (integer) maximum segment priority value: valid values are non-negative integers

19 COLOUR MODEL: has 1 parameter:

P1: (index) colour model: valid values are

1 RGB
2 CIELAB
3 CIELUV
4 CMYK
5 RGB-related
>5 reserved for registered values.

20 COLOUR CALIBRATION: has 13 parameters

P1: (index) calibration selection, valid values are

1 unspecified
2 reference white only
3 reference white, matrix1
4 reference white, matrix1, lookup tables
5 reference white, matrix1, lookup tables, matrix2
6 reference white, matrix1, matrix2
7 lookup tables, matrix2
8 matrix2
9 reference white, grid locations + grid values
>9 reserved for registered values

P2: (real) reference white value X component

P3: (real) reference white value Y component

P4: (real) reference white value Z component

P5: (real) 3x3 RGB calibration matrix: Xr, Xg, Xb, Yr, Yg, Yb, Zr, Zg, Zb.

P6: (real) 3x3 ABC transformation matrix: Ra, Rb, Rc, Ga, Gb, Gc, Ba, Bb, Bc

P7: (integer) number of lookup table entries (=n), valid values are non-negative integers.

P8: (colour component) 2n red lookup table entries: R, R'.

P9: (colour component) 2n green lookup table entries: G, G'.

P10: (colour component) 2n blue lookup table entries: B, B'.

P11:(integer) number of grid locations (=m), valid values are non-negative integers.

P12:(direct colour list) m CMYK grid locations.

P13: (m*(3 real)) m XYZ grid locations, each being: CIEXYZ-X, CIEXYZ-Y, CIEXYZ-Z

21 FONT PROPERTIES: has a variable number of parameter 3-tuples (P1,P2,P3); each parameter 3-tuple contains

P1: (index) property indicator, valid values are

1 font index
2 standard version
3 design source
4 font family
5 posture
6 weight
7 proportionate width
8 included glyph collections
9 included glyphs
10 design size
11 minimum size
12 maximum size
13 design group
14 structure
>14 reserved for registered values

P2: (integer) priority, valid values are non-negative integers.

P3: (structured data record) property value record, each record contains a single member and is comprised of [data type indicator, data element count, data element(s)]. Valid values of the records are

[(integer: i_IX) (integer: 1) (index: font-index)]
| [(integer: i_I) (integer: 1) (integer: standard-version)]
| [(integer: i_SF) (integer: 1) (string fixed: design-source)]
| [(integer: i_SF) (integer: 1) (string fixed: font-family)]
| [(integer: i_IX) (integer: 1) (index: posture)]
| [(integer: i_IX) (integer: 1) (index: weight)]
| [(integer: i_IX) (integer: 1) (index: proportionate-width)]
| [(integer: i_IX) (integer: n) (included-glyph-collections)(n)]
| [(integer: i_UI32) (integer: m) (included-glyphs)(m)]
| [(integer: i_R) (integer: 1) (real: design-size)]
| [(integer: i_R) (integer: 1) (real: minimum-size)]
| [(integer: i_R) (integer: 1) (real: maximum-size)]
| [(integer: i_UI8) (integer: 3) (design-group)]
| [(integer: i_IX) (integer: 1) (index: structure)]

NOTE 1 i_XX in the above denotes the integer value of the 'data type indicator' for data type "XX" as assigned in annex C of ISO/IEC 8632-1. For example i_IX represents the designator for data type IX, which is assigned the value 2.

NOTE 2 See NOTE 17, Table 1, for additional SDR formatting requirements.

(index) font index, valid values are positive integers.

(integer) standard version, valid values are

1 for ISO/IEC 9541:1991, first version

(string fixed) design source

(string fixed) font family

(index) posture, valid values are

0 not applicable
1 upright

2 oblique
3 back slanted oblique
4 italic
5 back slanted italic
6 other
>6 reserved for registered values

(index) weight, valid values are

0 not applcable
1 ultra light
2 extra light
3 light
4 semi light
5 medium
6 semi bold
7 bold
8 extra bold
9 ultra bold
>9 reserved for registered values

(index) proportionate-width, valid values are

0 not applicable
1 ultra condensed
2 extra condensed
3 condensed
4 semi condensed
5 medium
6 semi expanded
7 expanded
8 extra expanded
9 ultra expanded
>9 reserved for registered values

(index list) included glyph collections: one or more character set indexes.

(index list) included glyphs: 1 or more AFII 32-bit glyph identifiers of type UI32.

(real) design size: valid values are positive reals.

(real) minimum size: valid values are positive reals.

(real) maximum size: valid values are positive reals.

(3 octets) design group: a 3-tuple of parameters of type octet, which respectively define the class, subclass, and specific group components of the design group.

(index) structure: valid values are

0 undefined or not applicable
1 solid
2 outline
>2 reserved for registered values

22 GLYPH MAPPING: has 6 parameters:

P1: (index) character set index

P2: (enumerated) basis set character set type: valid values are as for CHARACTER SET LIST.

P3: (string fixed) basis set designation sequence tail: valid values are as for CHARACTER SET LIST.

P4: (integer) octets per code (=m), valid values are positive integers.

P5: (index) glyph source, valid values are

1 afii registry of 4-byte glyph identifiers
>1 reserved for registered values

P6: (structured data record) glyph-code associations. For glyph source value 1: contains two members, a code list and a glyph-name list:

[(integer: i_UI8), (integer: n(m+1)), (n(UI8,mUI8): list of (run-count,m-byte-code))]
[(integer: i_UI32), (integer: n), (n(UI32: glyph-name))]

NOTE 3 The code list is a list of run length specifiers, (UI8,mUI8), where each specifier encodes a sequence of 1 or more character codes. The first octet is the run count. If the first octet of the specifier equals 1, then only the single explicitly specified m-octet code is encoded (m is the value of P4). If the first octet is greater than 1, then the m-octet code is the base of a run sequence. Each of code in the sequence is 1 greater than the previous code. The glyph-name sequence is "parallel" to the code sequence. The glyph names are associated with the corresponding codes, and when there is a run longer than 1 in the codes, there is also a run longer than 1 in the glyph names. Each glyph name in a run is 1 greater than its predecessor.

NOTE 4 See NOTE 17, Table 1, for additional SDR formatting requirements.

23 SYMBOL LIBRARY LIST: has a variable parameter list

P1-Pn: n symbol library names (string fixed), the first name in the list is assigned to index 1, the second to index 2, etc.

24 PICTURE DIRECTORY:has 2 parameters:

P1: (enumerated) location data type selector: valid values are

0 UI8
1 UI16
2 UI32

P2: list of 3-tuples consisting of:

Picture Identifier (string fixed)
Picture Location ([ldt]) offset, in octets, from the beginning of the metafile
Application Structure Directory Location ([ldt]) offset, in octets, from the beginning of the metafile

NOTE 5 [ldt] designates UI8, UI16, UI32 as selected by location data type selector parameter. The values of picture-location are the offsets in octets from the beginning of the metafile to the start of the associated BEGIN PICTURE element. The values of Application Structure Directory Location are the offsets in octets from the start of the metafile to the start of the APPLICATION STRUCTURE DIRECTORY element of the associated picture.

8.4 Picture descriptor elements

Table 5 — Encoding of picture descriptor elements

Element Class 2	Element Id	Parameter Type	Parameter List Length	Parameter Range
SCALING MODE	1	E,R (FP)	BE+BFP	{0,1},FPR
COLOUR SELECTION MODE	2	E	BE	{0,1}
LINE WIDTH SPECIFICATION MODE	3	E	BE	{0..3}
MARKER SIZE SPECIFICATION MODE	4	E	BE	{0..3}
EDGE WIDTH SPECIFICATION MODE	5	E	BE	{0..3}
VDC EXTENT	6	2P	2BP	VDCR
BACKGROUND COLOUR	7	CD	BCD	CCOR
DEVICE VIEWPORT	8	2VP	2BVP	VCR
DEVICE VIEWPORT SPECIFICATION MODE	9	E,R(FP)	BE+BFP	{0,1,2},FPR
DEVICE VIEWPORT MAPPING	10	3E	3BE	{0,1} {0,1,2} {0,1,2}
LINE REPRESENTATION	11	2IX, SS,CO	2BIX+ BSS+BCO	+IXR,IXR, +SSR,COR
MARKER REPRESENTATION	12	2IX, SS,CO	2BIX+ BSS+BCO	+IXR,IXR, ++SSR,COR
TEXT REPRESENTATION	13	2IX, E, 2R,CO	2BIX+ BE+ 2BR+BCO	+IXR, {0,1,2} RR,++RR,COR
FILL REPRESENTATION	14	IX, E,CO, 2IX	BIX+ BE+BCO+ 2BIX	+IXR, {0..6},COR, IXR,+IXR
EDGE REPRESENTATION	15	2IX SS,CO	2BIX BSS+BCO	IXR,+IXR ++SSR,COR
INTERIOR STYLE SPECIFICATION MODE	16	E	BE	{0..3}
LINE AND EDGE TYPE DEFINITION	17	IX,SS,nI	BIX+BSS+nBI	- IXR,+SSR,++IR
HATCH STYLE DEFINITION	18	IX,E 4SS,SS I,nI, nIX	BIX+BE+ 4BSS+BSS+ BI+nBI+ nBIX	-IXR,{0,1}, ++SSR,+SSR +IR,++IR IXR
GEOMETRIC PATTERN DEFINITION	19	IX,N, 2P	BIX+BN+ 2BP	+IXR,NR VDCR
APPLICATION STRUCTURE DIRECTORY	20	E,n(SF,[ldt])	BE+n(BS+B[ldt])	{0,1,2}, (SR,[ldt]R,[ldt]R)

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

This parameter is always encoded as floating point, regardless of the value of the fixed/floating flag of REAL PRECISION. If a REAL PRECISION (floating, n, m) has preceded, then the precision used is n,m. If a REAL PRECISION element for floating point has not preceded, then a default precision of 9,23 (32-bit floating point) is used.

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
2 fractional
3 mm

4 MARKER SIZE SPECIFICATION MODE: has 1 parameter:

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

0 absolute
1 scaled
2 fractional
3 mm

5 EDGE WIDTH SPECIFICATION MODE: has 1 parameter:

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

0 absolute
1 scaled
2 fractional
3 mm

6 VDC EXTENT: has 2 parameters:

P1: (point) first corner

P2: (point) second corner

7 BACKGROUND COLOUR: has 1 parameter:

P1: (direct colour) background colour.

8 DEVICE VIEWPORT: has 2 parameters:

P1: (viewport point) first corner

P2: (viewport point) second corner

9 DEVICE VIEWPORT SPECIFICATION MODE: has 2 parameters:

P1: (enumerated) VC specifier: valid values are

0 fraction of drawing surface
1 millimetres with scale factor
2 physical device coordinates

P2: (real) metric scale factor, ignored if P1=0 or P1=2

This parameter is always encoded as floating point, regardless of the value of the fixed/floating flag of REAL PRECISION. If a REAL PRECISION (floating, n, m) has preceded, then the precision used is n,m. If a REAL PRECISION element for floating point has not preceded, then a default precision of 9,23 (32-bit floating point) is used.

10 DEVICE VIEWPORT MAPPING: has 3 parameters:

P1: (enumerated) isotropy flag: valid values are

0 not forced
1 forced

P2: (enumerated) horizontal alignment flag: valid values are

0 left
1 centre
2 right

P3: (enumerated) vertical alignment flag: valid values are

0 bottom
1 centre
2 top

11 LINE REPRESENTATION: has 4 parameters:

P1: (index) line bundle index

P2: (index) line type: valid values are

1 solid
2 dash
3 dot
4 dash-dot
5 dash-dot-dot
>5 reserved for registered values
negative for private use

P3: (size specification) line width: see Part 1, subclause 7.1 for its form.

P4: (colour) line colour: its form depends on COLOUR SELECTION MODE.

12 MARKER REPRESENTATION: has 4 parameters:

P1: (index) marker bundle index

P2: (index) marker type: valid values are

1 dot
2 plus
3 asterisk
4 circle
5 cross
>5 reserved for registered values
negative for private use

P3: (size specification) marker size: see Part 1, subclause 7.1 for its form.

P4: (colour) marker colour: its form depends on COLOUR SELECTION MODE.

13 TEXT REPRESENTATION: has 6 parameters:

P1: (index) text bundle index

P2: (index) text font index

P3: (enumerated) text precision: valid values are

0 string
1 character
2 stroke

P4: (real) character spacing

P5: (real) character expansion factor

P6: (colour) text colour; its form depends on COLOUR SELECTION MODE

14 FILL REPRESENTATION: has 5 parameters:

P1: (index) fill area bundle index

P2: (enumerated) interior style: valid values are

0 hollow
1 solid
2 pattern
3 hatch
4 empty
5 geometric pattern
6 interpolated

P3: (colour) fill colour: its form depends on COLOUR SELECTION MODE

P4: (index) hatch index: the following values are standardized:

1 horizontal
2 vertical
3 positive slope
4 negative slope
5 horizontal/vertical crosshatch
6 positive/negative slope crosshatch
>6 reserved for registered values
negative for private use

P5: (index) pattern index

15 EDGE REPRESENTATION: has 4 parameters:

P1: (index) edge bundle index

P2: (index) edge type: the following values are standardized:

1 solid
2 dash
3 dot
4 dash-dot
5 dash-dot-dot
>5 reserved for registered values
negative for private use

P3: (size specification) edge width: see Part 1, subclause 7.1 for its form.

P4: (colour) edge colour: its form depends on COLOUR SELECTION MODE.

16 INTERIOR STYLE SPECIFICATION MODE: has 1 parameter:

P1: (enumerated) valid values are

0 absolute
1 scaled
2 fractional
3 mm

17 LINE AND EDGE TYPE DEFINITION: has a variable parameter list:

P1: (index) line type, valid values are negative.

P2: (size specification) dash cycle repeat length: see Part 1, subclause 7.1 for its form.

P3-P(n+2): (integer) list of n dash elements

18 HATCH STYLE DEFINITION: has a variable parameter list:

P1: (index) hatch index, valid values are negative.

P2: (enumerated) style indicator: valid values are

0 parallel
1 cross hatch

P3: (4(size specification)) hatch direction vectors specifier (x,y,x,y): see Part 1, subclause 7.1 for its form.

P4: (size specification) duty cycle length: see Part 1, subclause 7.1 for its form.

P5: (integer) number of hatch lines (=n)

P6-P(5+n): (integers) list of n gap widths

P(6+n)-P(5+2n): (integers) list of n line types

19 GEOMETRIC PATTERN DEFINITION: has 4 parameters

P1: (index) geometric pattern index

P2: (name) segment identifier

P3: (point) first corner point

P4: (point) second corner point

20 APPLICATION STRUCTURE DIRECTORY:has 2 parameters

P1: (enumerated) location data type selector: valid values are

0 UI8
1 UI16
2 UI32

P2: list of pairs consisting of:

Application Structure Identifier (string fixed)
Application Structure Location ([Idt]) offsets, in octets, from the beginning of the picture containing the APS

NOTE - [Idt] designates UI8, UI16, UI32 as selected by location data type selector parameter. The values of Application Structure Location are the offsets in octets from the beginning of the BEGIN PICTURE element to the start of the associated BEGIN APPLICATION STRUCTURE element.

8.5 Control elements

Table 6 — Encoding of control elements

Element Class 3	Element Id	Parameter Type	Parameter List Length	Parameter Range
VDC INTEGER PRECISION	1	I	BI	16,24,32
VDC REAL PRECISION	2	E,2I	BE+2BI	{0,1}, {9,12,16,32}, {23,52,16,32}
AUXILIARY COLOUR	3	CO	BCO	COR
TRANSPARENCY	4	E	BE	{0,1}
CLIP RECTANGLE	5	2P	2BP	VDCR
CLIP INDICATOR	6	E	BE	{0,1}
LINE CLIPPING MODE	7	E	BE	{0,1,2}
MARKER CLIPPING MODE	8	E	BE	{0,1,2}
EDGE CLIPPING MODE	9	E	BE	{0,1,2}
NEW REGION	10	n/a	0	n/a
SAVE PRIMITIVE CONTEXT	11	N	BN	NR
RESTORE PRIMITIVE CONTEXT	12	N	BN	NR
PROTECTION REGION INDICATOR	17	2IX	2BIX	+IXR,{1,2,3}
GENERALIZED TEXT PATH MODE	18	E	BE	{0,1,2}
MITRE LIMIT	19	R	BR	++RR
TRANSPARENT CELL COLOUR	20	E,CO	BE+BCO	{0,1},COR

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

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 corner

P2: (point) second corner

6 CLIP INDICATOR: has 1 parameter:

P1: (enumerated) clip indicator: valid values are

0 off
1 on

7 LINE CLIPPING MODE: has 1 parameter:

P1: (enumerated) clipping mode: valid values are

0 locus
1 shape
2 locus then shape

8 MARKER CLIPPING MODE: has 1 parameter:

P1: (enumerated) clipping mode: valid values are

0 locus
1 shape
2 locus then shape

9 EDGE CLIPPING MODE: has 1 parameter:

P1: (enumerated) clipping mode: valid values are

0 locus
1 shape
2 locus then shape

10 NEW REGION: has no parameters

11 SAVE PRIMITIVE CONTEXT: has 1 parameter:

P1: (name) context name

12 RESTORE PRIMITIVE CONTEXT: has 1 parameter:

P1: (name) context name

17 PROTECTION REGION INDICATOR: has 2 parameters:

P1: (index) region index

P2: (index) region indicator: valid values are

1 off
2 clip
3 shield

18 GENERALIZED TEXT PATH MODE: has 1 parameter:

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

0 off
1 non-tangential
2 axis-tangential

19 MITRE LIMIT: has 1 parameter:

P1: (real) mitre limit

20 TRANSPARENT CELL COLOUR: has 2 parameters:

P1: (enumerated) transparency indicator, valid values are

0 off
1 on

P2: (colour) transparent cell colour specifier

8.6 Graphical primitive elements

Table 7 — Encoding of graphical primitive elements

Element Class 4	Element Id	Parameter Type	Parameter List Length	Parameter Range
POLYLINE	1	nP	nBP	VDCR
DISJOINT POLYLINE	2	nP	nBP	VDCR
POLYMARKER	3	nP	NBP	VDCR
TEXT	4	P,E,S	BP+BE+ BS	VDCR,{0,1}, SR
RESTRICTED TEXT	5	2VDC,P, E,S	2VDC+BP+ BE+BS	++VDCR,VDCR, {0,1},SR
APPEND TEXT	6	E,S	BE+BS	{0,1},SR
POLYGON	7	nP	nBP	VDCR
POLYGON SET	8	n(P,E)	n(BP+BE)	VDCR,{0..3}

Table 7 — Encoding of graphical primitive elements (continued)

Element Class 4	Element Id	Parameter Type	Parameter List Length	Parameter Range
CELL ARRAY	9	3P,3I, E,CLIST	3BP+3BI+ BE+nBCO	VDCR,+IR,++IR, {0,1},COR
GENERALIZED DRAWING PRIMITIVE	10	I,I,nP,D	2BI+nBP+ BS	IR,++IR, VDCR,SR
RECTANGLE	11	2P	2BP	VDCR
CIRCLE	12	P,VDC	BP+BVDC	VDCR, ++VDCR
CIRCULAR ARC POINT	13	3P	3BP	VDCR
CIRCULAR ARC 3 POINT CLOSE	14	3P,E	3BP+BE	VDCR,{0,1}
CIRCULAR ARC CENTRE	15	P,4VDC, VDC	BP+4BVDC+ BVDC	VDCR,VDCR, ++VDCR
CIRCULAR ARC CENTRE CLOSE	16	P,4VDC, VDC,E	BP+4BVDC+ BVDC+BE	VDCR,VDCR, ++VDCR,{0,1}
ELLIPSE	17	3P	3BP	VDCR
ELLIPTICAL ARC	18	3P,4VDC	3BP+4BVDC	VDCR,VDCR
ELLIPTICAL ARC CLOSE	19	3P,4VDC, E	3BP+4BVDC+ BE	VDCR,VDCR, {0,1}
CIRCULAR ARC CENTRE REVERSED	20	P,4VDC, VDC	BP+4BVDC+ BVDC	VDCR,VDCR, ++VDCR
CONNECTING EDGE	21	n/a	0	n/a
HYPERBOLIC ARC	22	3P,4VDC	3BP+4BVDC	VDCR
PARABOLIC ARC	23	3P	3BP	VDCR
NON-UNIFORM B-SPLINE	24	2I,nP, (n+m)R, 2R	2BI+nBP+ (n+m)BR+ 2BR	+IR,VDCR, ++RR, ++RR
NON-UNIFORM RATIONAL B-SPLINE	25	2I,nP, (n+m)R, 2R, nR	2BI+nBP+ (n+m)BR+ 2BR+ nBR	+IR,VDCR, ++RR, ++RR, ++RR
POLYBEZIER	26	IX,4nP(or) (3n+1)P	BIX+4nBP(or) BIX+(3n+1)P	{1,2}, VDCR
POLYSYMBOL	27	IX,nP	BIX+nBP	+IXR,VDCR
BITONAL TILE	28	IX,I, 2CO, SDR,BS	BIX+BIX+ 2BCO+ BSDR,BBS	++IXR,++IR, COR, n/a,BSR
TILE	29	IX,I, I,SDR,BS	BIX+BI+ BI+BSDR+BB S	++IXR,++IR, ++IR,n/a,BSR

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. With the exception of the first run of a row, the integer count of each run immediately follows the colour specifier of the preceding run with no intervening padding.

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

20 CIRCULAR ARC CENTRE REVERSED: 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

21 CONNECTING EDGE: has no parameters

22 HYPERBOLIC ARC: has 7 parameters:

P1: (point) centre point

P2: (point) transverse radius end point

P3: (point) conjugate radius end point

P4: (vdc) start vector x component

P5: (vdc) start vector y component

P6: (vdc) end vector x component

P7: (vdc) end vector y component

23 PARABOLIC ARC: has 3 parameters:

P1: (point) tangent intersection point

P2: (point) start point

P3: (point) end point

24 NON-UNIFORM B-SPLINE: has a variable parameter list:

P1: (integer) spline order (=m)

P2: (integer) number of control points (=n)

P(3)-P(2+n): (points) array of control points

P(3+n)-P(2+2n+m): (real) list of knots, of length n+m.

P(3+2n+m): (real) parameter start value

P(4+2n+m): (real) parameter end value

25 NON-UNIFORM RATIONAL B-SPLINE: has a variable parameter list:

P1: (integer) spline order (=m)

P2: (integer) number of control points (=n)

P(3)-P(2+n): (points) array of control points

P(3+n)-P(2+2n+m): (real) list of knots, of length n+m.

P(3+2n+m): (real) parameter start value

P(4+2n+m): (real) parameter end value

P(5+2n+m)-P(4+3n+m): (real) list of weights, of length n.

26 POLYBEZIER: has a variable parameter list:

P1: (index) continuity indicator: valid values are

1: discontinuous
2: continuous
>2 reserved for registered values

P2-Pn: (point) list of point sequences: each sequence defines a single bezier curve and contains 4 or 3 points according to the continuity indicator values 1 or 2, respectively (if the indicator is 2, the first curve, and only the first, is defined by 4 points).

27 POLYSYMBOL: has a variable parameter list:

P1: (index) symbol index

P2-P(n+1): (point) n symbol position points.

28 BITONAL TILE: has 6 parameters:

P1: (index) compression type: valid values are

0 null background
1 null foreground
2 T6
3 1-dimensional
4 T4 2-dimensional
5 bitmap (uncompressed)
6 run length
>6 reserved for registered values

P2: (integer) row padding indicator: valid values are non-negative integers.

P3: (colour) cell background colour

P4: (colour) cell foreground colour

P5: (structured data record) method-specific parameters, valid values are

[null_SDR], for compression types 1-5,
[(integer: i_l), (integer: 1), (integer: run-count precision)], for type=6,
as defined in the Register, for type>6.

Note 1 See NOTE 17, Table 1, for additional SDR formatting requirements.

P6 (bitstream) compressed cell colour specifiers

29 TILE: has 5 parameters:

P1: (index) compression type: valid values are

0 null background
1 null foreground
2 T6
3 T4 1-dimensional
4 T4 2-dimensional
5 bitmap (uncompressed)
6 run length
>6 reserved for registered values

P2: (integer) row padding indicator: valid values are non-negative integers.

P3: (integer) cell colour precision: valid values are as for the local colour precision of CELL ARRAY for compression types 0 - 5, or any value specified in the Register for compression type>6.

P4 (structured data record) method-specific parameters, valid values are

[null_SDR], for compression types 1-5,
[(integer: i_l), (integer: 1), (integer: run-count precision)], for type=6,
as defined in the Register, for type>6.

NOTE 2 See NOTE 17, Table 1, for additional SDR formatting requirements.

P5: (bitstream) compressed cell colour specifiers

8.7 Attribute elements

Table 8 — Encoding of attribute elements

Element Class 5	Element Id	Parameter Type	Parameter List Length	Parameter Range
LINE BUNDLE INDEX	1	IX	BIX	+IXR
LINE TYPE	2	IX	BIX	IXR
LINE WIDTH	3	SS	BSS	++SSR
LINE COLOUR	4	CO	BCO	COR
MARKER BUNDLE INDEX	5	IX	BIX	+IXR
MARKER TYPE	6	IX	BIX	IXR
MARKER SIZE	7	SS	BSS	++SSR
MARKER COLOUR	8	CO	BCO	COR
TEXT BUNDLE INDEX	9	IX	BIX	+IXR
TEXT FONT INDEX	10	IX	BIX	+IXR
TEXT PRECISION	11	E	BE	{0..2}
CHARACTER EXPANSION FACTOR	12	R	BR	++RR
CHARACTER SPACING	13	R	BR	RR
TEXT COLOUR	14	CO	BCO	COR
CHARACTER HEIGHT	15	VDC	BVDC	++VDCR
CHARACTER ORIENTATION	16	4VDC	4BVDC	VDCR
TEXT PATH	17	E	BE	{0..3}
TEXT ALIGNMENT	18	2E, R,R	2BE+ 2BR	{0..4}, {0..6}, 2RR
CHARACTER SET INDEX	19	IX	BIX	+IXR
ALTERNATE CHARACTER SET INDEX	20	IX	BIX	+IXR
FILL BUNDLE INDEX	21	IX	BIX	+IXR
INTERIOR STYLE	22	E	BE	{0..6}
FILL COLOUR	23	CO	BCO	COR
HATCH INDEX	24	IX	BIX	IXR
PATTERN INDEX	25	IX	BIX	+IXR
EDGE BUNDLE INDEX	26	IX	BIX	+IXR
EDGE TYPE	27	IX	BIX	IXR
EDGE COLOUR	29	CO	BCO	COR
EDGE VISIBILITY	30	E	BE	{0,1}
FILL REFERENCE POINT	31	P	BP	VDCR

Table 8 — Encoding of attribute elements (continued)

Element Class 5	Element Id	Parameter Type	Parameter List Length	Parameter Range
PATTERN TABLE	32	IX,3I, nx*nyCO	BIX+3BI+ nx*nyBCO	+IXR,+IR, ++IR,COR
PATTERN SIZE	33	4SS	4BSS	SSR
COLOUR TABLE	34	CI,nCD	BCI+nBCD	CIR,CCOR
ASPECT SOURCE FLAGS	35	n(E,E)	n(2BE)	{0..17},{0,1}
PICK IDENTIFIER	36	N	BN	NR
LINE CAP	37	IX,IX	2BIX	+IXR
LINE JOIN	38	IX	BIX	+IXR
LINE TYPE CONTINUATION	39	IX	BIX	+IXR
LINE TYPE INITIAL OFFSET	40	R	BR	++RR
TEXT SCORE TYPE	41	n(IX,E)	nBIX+nBE	IXR,{0,1}
RESTRICTED TEXT TYPE	42	IX	BIX	+IXR
INTERPOLATED INTERIOR	43	IX,2nSS, I,mR,kCO	2BIX+2nBSS+ BI+mBR+kBCO	{1..3},SSR, +IR,RR,COR
EDGE CAP	44	IX,IX	2BIX	+IXR
EDGE JOIN	45	IX	BIX	+IXR
EDGE TYPE CONTINUATION	46	IX	BIX	+IXR
EDGE TYPE INITIAL OFFSET	47	R	BR	++RR
SYMBOL LIBRARY INDEX	48	IX	BIX	+IXR
SYMBOL COLOUR	49	CO	BCO	COR
SYMBOL SIZE	50	E,2VDC	BE+2BVDC	{0..2},VDCR
SYMBOL ORIENTATION	51	4VDC	4BVDC	VDCR

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
>5 reserved for registered values
negative for private use

3 LINE WIDTH: has 1 parameter:

P1: (size specification) line width: see Part 1, subclause 7.1 for its form.

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

P1: (colour) line colour

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
>5 reserved for registered values
negative for private use

7 MARKER SIZE: has 1 parameter:

P1: (size specification) marker size: see Part 1, subclause 7.1 for its form.

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

P1: (colour) marker colour

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

15 CHARACTER HEIGHT: has 1 parameter:

P1: (vdc) character height.

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 are:

0 normal horizontal
1 left
2 centre
3 right
4 continuous horizontal

P2: (enumerated) vertical alignment

0 normal vertical
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
5 geometric pattern
6 interpolated

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

P1: (colour) fill colour

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 horizontal/vertical crosshatch
6 positive/negative slope crosshatch
>6 reserved for registered values
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
>5 reserved for registered values
negative for private use

28 EDGE WIDTH: has 1 parameter:

P1: (size specification) edge width: see part 1, subclause 7.1 for its form.

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

P1: (colour) edge colour

30 EDGE VISIBILITY: has 1 parameter:

P1: (enumerated) edge visibility: valid values are

0 off
1 on

31 FILL REFERENCE POINT: has 1 parameter:

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 as for the local colour precision parameter of CELL ARRAY.

P5: (colour array) pattern definition

33 PATTERN SIZE: has 4 parameters:

P1: (size specification) pattern height vector, x component: see part 1, subclause 7.1 for its form.

P2: (size specification) pattern height vector, y component: see part 1, subclause 7.1 for its form.

P3: (size specification) pattern width vector, x component: see part 1, subclause 7.1 for its form.

P4: (size specification) pattern width vector, y component: see part 1, subclause 7.1 for its form.

NOTE Pattern size may only be 'absolute' (VDC) in Version 1 and 2 metafiles. In Version 3 and 4 metafiles it may be expressed in any of the modes which can be selected with INTERIOR STYLE SPECIFICATION MODE.

34 COLOUR TABLE: has 2 parameters:

P1: (colour index) starting colour table index

P2: (direct colour list) list of direct colour values (>3-tuples or 4-tuples of

direct colour components (CCO))

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

36 PICK IDENTIFIER: has 1 parameter:

P1: (name) pick identifier

37 LINE CAP: has 2 parameters:

P1: (index) line cap indicator: the following values are standardized:

1 unspecified
2 butt
3 round
4 projecting square
5 triangle
>5 reserved for registered values

P2: (index) dash cap indicator: valid values are

1 unspecified
2 butt
3 match
>3 reserved for registered values

38 LINE JOIN: has 1 parameter:

P1: (index) line join indicator: the following values are standardized:

1 unspecified
2 mitre
3 round
4 bevel
>4 reserved for registered values

39 LINE TYPE CONTINUATION: has 1 parameter:

P1: (index) continuation mode: the following values are standardized:

1 unspecified
2 continue
3 restart
4 adaptive continue
>4 reserved for registered values

40 LINE TYPE INITIAL OFFSET: has 1 parameter:

P1: (real) line pattern offset

41 TEXT SCORE TYPE: has 1 parameter:

P1-Pn: list of score type, score indicator pairs (index,enumerated): the following values are standardized for the score type:

1 right score
2 left score
3 through score
4 kendot
>4 reserved for registered values

valid values for the score indicators are

0 off
1 on

42 RESTRICTED TEXT TYPE: has 1 parameter:

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

1 basic
2 boxed-cap
3 boxed-all
4 isotropic-cap
5 isotropic-all
6 justified
>6 reserved for registered values

43 INTERPOLATED INTERIOR: has a variable parameter list:

P1: (index) style: valid values are

1 parallel
2 elliptical
3 triangular
>3 reserved for registered values

P2: (2n(size specification)) reference geometry: see part 1, subclause 7.1 for its form.

P3: (integer) number of stages (=m)

P4: (real) array of m stage designators

P5: (colour) array of k colour specifiers: k=3 for triangular, m+1 otherwise.

44 EDGE CAP: has 2 parameters:

P1: (index) edge cap indicator: the following values are standardized:

1 unspecified
2 butt
3 round
4 projected square
5 triangle
>5 reserved for registered values

P2: (index) dash cap indicator: valid values are

1 unspecified
2 butt
3 match
>3 reserved for registered values

45 EDGE JOIN: has 1 parameter:

P1: (index) edge join indicator: the following values are standardized:

1 unspecified
2 mitre
3 round
4 bevel
>4 reserved for registered values

46 EDGE TYPE CONTINUATION: has 1 parameter:

P1: (index) continuation mode: the following values are standardized:

1 unspecified
2 continue
3 restart
4 adaptive continue
>4 reserved for registered values

47 EDGE TYPE INITIAL OFFSET: has 1 parameter:

P1: (real) edge pattern offset

48 SYMBOL LIBRARY INDEX: has 1 parameter:

P1: (index) symbol library index

49 SYMBOL COLOUR: has 1 parameter:

P1: (colour) symbol colour

50 SYMBOL SIZE: has 3 parameters:

P1: (enumerated) scale indicator: valid values are

0 height
1 width
2 both

P2: (vdc) symbol height

P2: (vdc) symbol width

51 SYMBOL ORIENTATION: has 4 parameters:

P1: (vdc) up vector x component

P2: (vdc) up vector y component

P3: (vdc) base vector x component

P4: (vdc) base vector y component

8.8 Escape element

Table 9 — Encoding of escape element

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

Additional description of the element 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.

8.9 External elements

Table 10 — Encoding of external elements

Element Class 7	Element Id	Parameter Type	Parameter List Length	Parameter Range
MESSAGE	1	E,SF	BE+BS	{0,1},SR
APPLICATION DATA	2	I,D	BI+BS	IR,SR

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 fixed) 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.

8.10 Segment control and segment attribute elements

Table 11 — Encoding of segment elements

Element Class 8	Element Id	Parameter Type	Parameter List Length	Parameter Range
COPY SEGMENT	1	N,4R, 2VDC, E	BN+4BR+ 2BVDC + BE	NR,RR, VDCR, {0,1}
INHERITANCE FILTER	2	nE,E	(n+1)BE	{0..86},{0,1}
CLIP INHERITANCE	3	E	BE	{0,1}
SEGMENT TRANSFORMATION	4	N,4R, 2VDC	BN+4BR+ 2BVDC	NR,RR, VDCR
SEGMENT HIGHLIGHTING	5	N,E	BN+BE	NR,{0,1}
SEGMENT DISPLAY PRIORITY	6	N,I	BN+BI	NR,++IR
SEGMENT PICK PRIORITY	7	N,I	BN+BI	NR,++IR

Additional description of the elements in Table 11:

Code Description

1 COPY SEGMENT: has 3 parameters:

P1: (name) segment identifier

P2: The next 6 values are components of a transformation matrix consisting of a scaling and rotation portion (2 x 2 R) and a translation portion (2 x 1 VDC). In the binary encoding this is expressed as a 2 x 3 matrix of the form:

a11: (real) x scale component
a12: (real) x rotation component
a21: (real) y rotation component
a22: (real) y scale component
a13: (vdc) x translation component
a23: (vdc) y translation component

P3: (enumerated) segment transformation application: valid values are

0: no
1: yes

2 INHERITANCE FILTER: has two parameters. The first is a list of attribute or group designators. The second is a single setting value.

P1: (enumerated list) list of one or more of:

0 line bundle index
1 line type
2 line width
3 line colour
4 line clipping mode
5 marker bundle index
6 marker type
7 marker size

8 marker colour
9 marker clipping mode
10 text bundle index
11 text font index
12 text precision
13 character expansion factor
14 character spacing
15 text colour
16 character height
17 character orientation
18 text path
19 text alignment
20 fill bundle index
21 interior style
22 fill colour
23 hatch index
24 pattern index
25 edge bundle index
26 edge type
27 edge width
28 edge colour
29 edge visibility
30 edge clipping mode
31 fill reference point
32 pattern size
33 auxiliary colour
34 transparency
35 line attributes
36 marker attributes
37 text presentation and placement attributes
38 text placement and orientation attributes
39 fill attributes
40 edge attributes
41 pattern attributes
42 output control
43 pick identifier
44 all attributes and control
45 all
46 line type asf
47 line width asf
48 line colour asf
49 marker type asf
50 marker size asf
51 marker colour asf
52 text font index asf
53 text precision asf
54 character expansion factor asf
55 character spacing asf
56 text colour asf
57 interior style asf
58 fill colour asf
59 hatch index asf
60 pattern index asf
61 edge type asf
62 edge width asf
63 edge colour asf
64 line asfs
65 marker asfs
66 text asfs
67 fill asfs

68 edge asfs
69 all asfs
70 mitre limit
71 line cap
72 line join
73 line type continuation
74 line type initial offset
75 text score type
76 restricted text type
77 interpolated interior
78 edge cap
79 edge join
80 edge type continuation
81 edge type initial offset
82 symbol library index
83 symbol colour
84 symbol size
85 symbol orientation
86 symbol attributes

P2: (enumerated) setting: valid values are

0 state list
1 segment

3 CLIP INHERITANCE: has 1 parameter

P1: (enumerated) clip inheritance: valid values are

0 state list
1 intersection

4 SEGMENT TRANSFORMATION: has 2 parameters:

P1: (name) segment identifier

P2: The next 6 values are components of a transformation matrix consisting of a scaling and rotation portion (2 x 2 R) and a translation portion (2 x 1 VDC). In the binary encoding this is expressed as a 2 x 3 matrix of the form:

a11: (real) x scale component
a12: (real) x rotation component
a21: (real) y rotation component
a22: (real) y scale component
a13: (vdc) x translation component
a23: (vdc) y translation component

5 SEGMENT HIGHLIGHTING: has 2 parameters:

P1: (name) segment identifier

P2: (enumerated) highlighting: valid values are

0 normal
1 highlighted

6 SEGMENT DISPLAY PRIORITY: has 2 parameters:

P1: (name) segment identifier

P2: (integer) segment display priority: valid values are non-negative integers