INTERNATIONAL STANDARD

ISO 8632-4

First edition
1987-08-01

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

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

Part 4 :
Clear text encoding

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

Partie 4 : Codage en clair des textes

Reference number
ISO 8632-4 : 1987 (E)

Foreword

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

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

International Standard ISO 8632-4 was prepared by Technical Committee ISO/TC 97, *Information processing systems*.

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

Printed in Switzerland

CONTENTS

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

Part 4 :
Clear text encoding

0 Introduction

0.1 Purpose of the clear text encoding

The Clear Text Encoding of the Computer Graphics Metafile (CGM) provides a representation of the Metafile syntax that is easy to type, edit and read. It allows a metafile to be edited with any standard text editor, using the internal character code of the host computer system.

0.2 Primary objectives

a. HUMAN EDITABLE: The Clear Text Encoding should be able to be hand edited or, if desired, hand constructed.

b. HUMAN-FRIENDLY: The Clear Text Encoding should be easy and natural for people to read and edit. Although what is easiest and most natural is a subjective judgment that varies among users, contributing factors such as ease of recognition, ease of remembering, avoidance of ambiguity, and prevention of mistyping have all been considered.

c. MACHINE-READABLE: The Clear Text Encoding should be able to be parsed by software.

d. Suitable for USE IN A WIDE VARIETY OF EDITORS: The Clear Text Encoding should not have any features that make it difficult to edit in normal text editors.

e. Facilitate INTERCHANGE BETWEEN DIVERSE SYSTEMS: The Clear Text Encoding should be encoded in such a way as to maximize the set of systems which can utilize it. No assumptions should be made as to word size or arithmetic modes used to interpret the metafile.

f. Use STANDARDIZED ABBREVIATIONS as much as possible. Where language encoding of other graphics standards have established standard abbreviations, or where common practice in the data processing and graphics industries has established well known abbreviations, these abbreviations are used. In accordance with the principle of "least astonishment", this approach should minimize the time needed to learn to use this encoding.

0.3 Secondary objectives

Because other CGM encodings are targeted toward CPU efficiency (CGM Binary Encoding) and information density (CGM Character Encoding), these objectives are considered of secondary importance for the CGM Clear Text Encoding.

0.4 Relationship to other International Standards

The set of characters required to implement the Clear Text Encoding is a subset of those included in national versions of ISO 646. Any character set that can be mapped to and from that subset may be used to implement the encoding.

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 part 1, 4.11.)

0.5 Status of annexes

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

1 Scope and field of application

This part of ISO 8632/4 specifies a clear text encoding of the Computer Graphics Metafile. For each of the elements specified in ISO 8632/1, a clear text encoding is specified. Allowed abbreviations are specified. The overall format of the metafile and the means by which comments may be interspersed in the metafile is specified.

This encoding of the CGM allows metafiles to be created and maintained in a form which is simple to type, easy to edit and convenient to read.

2 References

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

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

3 Notational conventions

Unbracketed strings are terminals of this grammar which appear exactly, subject to the notes on case and null characters given below.

Bracketed strings are either non-terminals (with further productions given), character symbol names (such as COMMA), or parameters of the CGM element in the form <x:y> (see ISO 8632/1 for further explanation of these items).

"::=" is read as "becomes" or "is realized as".

<...>* = star closure (0 or more occurrences).
<...>+ = plus closure (1 or more occurrences).
<...>o = optional (exactly 0 or 1 occurrences).
<x:y> = parameter type x with meaning y
<x|y> = exactly one of x or y
{...} = a comment (not part of the production)

SPACES are used for readability in the grammar description; SPACES in the actual metafile are indicated through the separator productions given below.

The metasymbols used in describing the grammar do not appear in the actual metafile.

4 Entering and leaving the metafile environment

4.1 Generic clear text and instantiations

The Clear Text Encoding is described in a generic fashion that permits it to be used with any character set capable of representing those characters enumerated in the Character Repertoire (see part 1, 4.7.6). An instantiation of the Clear Text Encoding is specified by defining the character set and coding technique to be used (for example, standard national character sets based on ISO 646, non-standard character sets such as EBCDIC, etc).

It is recommended that an instantiation of the Clear Text Encoding bound to the standard national character set based on ISO 646 be used in order to maximize portability of Clear Text metafiles between diverse systems. This also provides an encoding which can be incorporated into an ISO 2022 text environment as a complete code, to permit intermixing of text and graphics for applications which place a high priority on human readability.

4.2 Implicitly entering the metafile environment

The Clear Text coding environment may be entered implicitly by agreement between the interchanging parties. This is suitable only if there is not to be any interchange with services using other coding techniques, and if it is known by prior agreement which instantiation of the syntax is being used.

4.3 Designating and invoking the CGM coding environment from ISO 2022

For interchange with services using the code extension techniques of ISO 2022, the (standard national version) ISO 646 instantiation of the CGM Clear Text Encoding may be designated and invoked from the ISO 2022 environment by the following escape sequence:

ESC 2/5 F

where ESC is the bit combination 1/11, and F refers to a bit combination that will be assigned by the ISO Registration Authority for ISO 2375.

The first bit combination occurring after this escape sequence will then represent the beginning of a CGM metafile element or one of the "soft separators" or "null characters" defined below.

The following escape sequence may be used to return to the ISO 2022 coding environment:

ESC 2/5 4/0

This not only returns to the ISO 2022 coding environment, but also restores the designation and invocation of coded character sets to the state that existed prior to entering the ISO 646 CGM coding environment with the ESC 2/5 F sequence. (The terms "designation" and "invocation" are defined in ISO 2022.)

It is permissable to make transitions between ISO 2022 and the metafile environment between pictures in the metafile as well as between metafiles. The state of the metafile interpreter and the state of the ISO 2022 environment are maintained separately and not stacked. The state of the metafile interpreter before BEGIN METAFILE or after END METAFILE is undefined, and sending a picture without a preceding BEGIN METAFILE and metafile descriptor is nonconforming interchange.

5 Metafile format

A metafile in the Clear Text Encoding consists of a stream of characters forming a series of elements, each of which starts with an element name and ends with one of the element delimiters, either the SLASH character (also known as SLANT or SOLIDUS) or the SEMICOLON character. These characters do not act as element delimiters when occurring within the bounds of a string parameter, as defined below.

5.1 Character repertoire

In order to achieve objective (e) of sub-clause 0.2, the character repertoire of the Clear Text Encoding will be limited to those characters enumerated below, except for string parameters, which may contain any characters from the repertoire described in ISO 8632/1 (see ISO 8632/1, 4.7.6).

- Upper-case characters:
  "A", "B", "C", "D", "E", "F", "G", "H", "I",
  "J", "K", "L", "M", "N", "O", "P", "Q", "R",
  "S", "T", "U", "V", "W", "X", "Y", "Z"
- Lower-case characters: (see note 1)
  "a", "b", "c", "d", "e", "f", "g", "h", "i",
  "j", "k", "l", "m", "n", "o", "p", "q", "r",
  "s", "t", "u", "v", "w", "x", "y", "z"
- Digits:
  "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"
- " " (SPACE character)
- "+" (PLUS character)
- "-" (MINUS character)
- "#" (NUMBER SIGN)
- ";" (SEMICOLON character)
- "/" (SLASH, SLANT, or SOLIDUS character)
- "(" (LEFT or OPEN PARENTHESIS character)
- ")" (RIGHT or CLOSE PARENTHESIS character)
- "," (COMMA character)
- "." (DECIMAL POINT or PERIOD character)
- "'" (APOSTROPHE or SINGLE QUOTE character)
- """ (DOUBLE QUOTE character)
- "_" (UNDERSCORE character) (see note 2)
- "$" (DOLLAR SIGN or CURRENCY symbol) (see note 2)
- "%" (PERCENT SIGN character)

Lower-case characters are considered to be the same as upper-case characters, when occurring outside of string parameters. Any combination of lower-case and upper-case characters may be used within an element or enumerated parameter name.

The UNDERSCORE and DOLLAR SIGN symbols are defined as "null characters" within this encoding. They may appear anywhere within the metafile, and are mandated to have no effect on parsing (outside of string parameters). They are available for the generator or editor of the metafile to use in enhancing readability of tokens.

NOTE — To illustrate: the following are all equivalent: linetype, LINETYPE, LineType, line_type, $LINETYPE, L_I_N_E$T_Y_P_E; similarly, the following are all equivalent: 123456, $123456, 123_456, $123_456, $12$34$56.

Those control characters that are format effectors (BACKSPACE, CARRIAGE RETURN, LINEFEED, NEWLINE, HORIZONTAL TAB, VERTICAL TAB, and FORMFEED) are permitted in the metafile, but are treated as SPACE characters (that is, as soft delimiters) by the metafile interpreter whenever they occur outside of a string parameter. They may be used to assist in formatting the metafile to

improve its readability. (The effect of such format effectors within string parameters is as defined in ISO 8632/1.) A metafile written in the Clear Text Encoding is considered to be non-conforming interchange if it includes characters other than those listed in the repertoire and the format effectors (outside of string parameters). Implementation-dependent extensions which require use of characters other than the above should be embedded in the string parameters of the ESCAPE, MESSAGE, or APPLICATION DATA elements, or in comments.

The code set of the characters is not fixed by ISO 8632/4. In order to accomplish the objective of editability, it is permitted to encode the Clear Text Encoding using the character set codes native to the system. It is presumed that standard conversion facilities can be used in translating Clear Text CGM files from one system's character set codes to another, consistent with the treatment of other text files being transferred between systems. It is recommended that the ISO 646 codes be used to encode Clear Text metafiles for transport between diverse systems.

Null characters or format effectors outside of text strings which do not exist in the target system's encoding may be dropped in such translation, and lower-case letters translated to upper case as necessary, without altering the information content of the metafile. Likewise, the two statement delimiter characters are interchangeable and may be changed in such a translation without affecting the information content of the metafile. The two string delimiter characters are interchangeable, but any translation shall correctly handle the possible occurrence of either string delimiter character within the string parameter.

5.2 Separators

5.2.1 Element separators

<TERM> ::= <OPTSEP> <SLASH ¦ SEMICOLON> <OPTSEP>

The SEMICOLON and SLASH characters may be used interchangeably to delimit elements in a Clear Text metafile. These elements do not, however, terminate an element when they occur within a string parameter, as described below.

The elements of the metafile are not terminated by the ends of records, as indicated by control characters such as CR (carriage return) or LF (linefeed). Multiple elements may exist on one line, and any element may extend over multiple lines.

5.2.2 Parameter separators

The following productions are used in the Clear Text Encoding for parameter separators:

```
<SEPCHAR>    ::=  <SPACE ¦ CARRIAGE RETURN ¦ LINEFEED
                  ¦ HORIZONTAL TAB ¦ VERTICAL TAB ¦ FORMFEED>
<SOFTSEP>    ::=  <SEPCHAR>+
<OPTSEP>     ::=  <SEPCHAR>*
<HARDSEP>    ::=  <OPTSEP> <COMMA> <OPTSEP>
<SEP>        ::=  <SOFTSEP> ¦ <HARDSEP>
```

Most commands require a SOFTSEP after the element name (e.g., at least one space). This permits element names to be formed from a mixture of alpha and numeric characters.

The separator between parameters is usually a SEP. This format permits omission of parameters. (Two consecutive COMMAs indicate an omitted parameter.)

Since the enclosing APOSTROPHE or DOUBLE QUOTE character sufficiently delineates string parameters, and the statement delimiter SLASH also sets off the data on either side of it, the separators between these characters and adjacent parameters or element names are optional (OPTSEP).

SEPCHAR characters are not permitted within a name (element or enumerated type), or within the representation of a numeric parameter. Any place where a SEPCHAR is permitted (other than inside a

string parameter), an arbitrary number of SEPCHARs may be used.

5.2.3 Comments in the metafile

Comments may be included in a Clear Text metafile, to enhance its readability and usefulness. Some uses of comments might be to document hand-edited changes to the metafile, or as "notes to one's self" made while reading a metafile. To include other forms of nongraphical information in the metafile, it is suggested that the APPLICATION DATA element be used. If it is desired to convert a Clear Text metafile to one of the other encodings, comments may be either dropped or converted to APPLICATION DATA elements.

Comments are encoded as a series of printing characters and <SEPCHAR>s surrounded by "%" (PERCENT SIGN) characters. The text of the comment may not include this comment delimiter character.

Comments may be included any place that a separator may be used, and are equivalent to a <SOFTSEP>; they may be replaced by a SPACE character in parsing, without affecting the meaning of the metafile.

5.3 Encoding of parameter types

5.3.1 Integer-bound types

INTEGERS, INTEGER COORDINATES, INDICES, and the components of COLOUR DIRECT parameters are all bound to signed integers, indicated in the encoding as I.

<I> ::= <decimal integer> | <based integer>

<decimal integer> ::= <sign>o <digit>+

<sign> ::= <PLUS SIGN> | <MINUS SIGN>

<based integer> ::= <sign>o <base> <NUMBER SIGN> <extended digit>+

<base> ::= 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<extended digit> ::= <digit> | A | B | C | D | E | F | a | b | c | d | e | f

The null characters are permitted within numbers, but are not shown in the productions for simplicity.

A decimal integer has an optional sign and at least one digit. If the sign appears, it immediately precedes the number with no intervening SPACE (or other <SEPCHAR>) characters allowed.

A based integer has an optional sign, a base (an unsigned integer in the range 2..16 inclusive, represented in base 10), a "#", and a string of one or more extended digits. If the sign appears, it immediately precedes the number with no intervening SPACE (or other <SEPCHAR>) characters allowed. The extended digits used shall be valid for the base named or the metafile is not conforming; e.g., for base 8 the digits "8", "9", etc. are not valid, for base 2 only the digits "0" and "1" are valid, and so forth. Case is not significant for the extended digits.

If the sign is omitted for either form, the number is considered non-negative.

Both the base and the <extended digit>+ are interpreted as unsigned numbers, and the final result negated if a MINUS SIGN preceded the number. No assumptions should be made as to the word size of the metafile interpreter, or whether the underlying arithmetic is one's complement, two's complement, or sign-magnitude. For example, -1 would be encoded in hexadecimal as -16#1, -16#0001, etc. rather than 16#FFFF. Of course, metafiles may be created utilizing prior knowledge of the intended target machine, but any such assumptions will limit the portability of the metafile and are discouraged.

Examples:

0, 007, -5, +123_456
The following are equivalent: 65535, 16#FFFF, 16#ffff, 8#177777, 2#1111_1111_1111_1111
The following are equivalent: -32_768, -16#8000, -8#100000, -2#10000000_00000000

Interpretation of numerically bound parameters will be "free field", that is, there is an implied radix point to the right of the rightmost digit, and neither leading nor trailing spaces are significant. Leading zeroes are not significant.

5.3.2 Real-bound types

REALS and REAL COORDINATES are bound to real numbers, indicated in the encoding as R. These are written as either explicit-point numbers or scaled-real numbers (or decimal integers, where appropriate).

```
<R>                          ::= < explicit-point number > ¦
                                 < scaled-real number> ¦
                                 < decimal integer >

< explicit-point number >    ::= <sign>o
                                 <
                                 <<digit>+ <PERIOD> <digit>*>
                                 ¦
                                 <<digit>* <PERIOD> <digit>+>
                                 >

< scaled-real number>        ::= <body> < E ¦ e > <exponent>

<body>                       ::= <explicit-point number> ¦
                                 <decimal integer>

<exponent>                   ::= <decimal integer>
```

The interpretation of the scaled-real number is the same as standard scientific notation (similar to FORTRAN "E" format), where the number represented by <body> is multiplied by 10 taken to the power <exponent>.

There shall be at least one digit in an explicit-point number and in the body of a scaled-real number, which in the case of a single-digit number may appear on either side of the radix point. It is recommended but not required that there be at least one digit before the radix point, for numbers with only a fractional part. Zero may be encoded as "0.", ".0", "0.0", "0", etc., although the second form is not recommended.

In the case of a scaled-real number (one where an "E" or "e" appears), at least one digit shall appear in the <exponent>. No SPACE or other <SEPCHAR> characters are permitted between the <body> and the "E" or "e", or between the "E" or "e" and the <exponent>.

The interpretation of parameters bound to this data type will be "free field", that is, if there is an explicit radix point, it sets the radix point of the internal representation, and neither leading nor trailing spaces or zeroes are significant. If the radix point is omitted, it is implied to be at the right of the rightmost digit of the explicit-point number or of the <body> of the scaled-real number. Thus, decimal I-format numbers are permitted to appear in a conforming metafile for parameters bound to real numbers when there is no fractional part.

For real numbers in all formats, the only permitted base of representation is base 10.

If the <sign> ("+" or "-") is omitted, the number is assumed to be non-negative. If the sign is present, it immediately precedes the body of the number, with no SPACE (or other <SEPCHAR>) characters allowed between it and the leftmost digit or radix point of the body of the number.

COMMA, SPACE and other <SEPCHAR> characters are not permitted within a number, but <NULLCHAR> characters are permitted (and have no effect on parsing).

Examples:

3.14159
7.853982E-7
271828e-5
42
-.04321 (not recommended form)
-0.043_21
42E2

5.3.3 String-bound types

STRING parameters are represented as character strings immediately surrounded by a matched pair of either APOSTROPHE (SINGLE QUOTE) or DOUBLE QUOTE characters.

If an APOSTROPHE is required in a string delimited with APOSTROPHES, it is represented by two adjacent APOSTROPHES at that position in the string. Likewise, if a DOUBLE QUOTE character is required in a string delimited with DOUBLE QUOTE characters, it is represented by two adjacent DOUBLE QUOTE characters. For example, the following are equivalent

```
TEXT 0 0 FINAL "Murphy's Law: ""If it can go wrong, it will.""";
TEXT 0 0 FINAL 'Murphy''s Law: "If it can go wrong, it will."';
```

DATA RECORD data type is represented as a string in this encoding.

STRING parameters are indicated in the encoding as S.

5.3.4 Enumerated types

Enumerated types are bound to names, similarly to the element names. Where an implementation wishes to support private enumerated type values, these shall be encoded as the letters "PRIV" followed by a string of <alpha ¦ digit ¦ null character>*.

5.3.5 Derived types

In addition to the I, R, and S parameter formats, the following abbreviations are used as shorthand for the productions shown.

VDC ::= <I> ¦ <R> { coordinate data, depending on VDC TYPE }

<RED GREEN BLUE> ::= <I:RED> <SEP> <I:GREEN> <SEP> <I:BLUE>

K ::= <I> ¦ <RED GREEN BLUE> { colour value, depending on COLOUR SELECTION MODE }

POINTREC ::= <VDC> <SEP> <VDC>

P ::= <POINTREC> ¦
< <LEFT PAREN> <OPTSEP> <POINTREC> <OPTSEP> <RIGHT PAREN> >

{ COORDINATE in VDC space. Parentheses are optional. If they are used, they shall group exactly two VDC numbers. The parenthesized form is intended to aid readability of the metafile. If there are not two numbers in each parenthesized group, the metafile is non-

conforming interchange. Any attempted error recovery or exception handling which a metafile interpreter may use in this situation is neither defined nor constrained by ISO 8632/4. A metafile interpreter need not use the parentheses in parsing; in this case, they are treated as SPACE characters rather than as NULL characters (i.e., they act as soft delimiters). }

V ::= <VDC> ¦ <R>

{ Used for line width, marker size, and edge width, this parameter type depends on the corresponding SPECIFICATION MODE element in the picture descriptor. }

5.4 Forming names

The approach was taken of selecting abbreviations for words used to name elements and enumeration types in the CGM, and concatenating them in order.

5.4.1 Words deleted

ANNOUNCER	FACTOR	SPECIFICATION
AREA	INDICATOR	
BUNDLE	REPLACEMENT	
CIRCULAR	SELECTION	

5.4.2 Words added

INCREMENTAL

5.4.3 Words used unabbreviated

ABSTRACT	END	PLUS
ACTION	ESCAPE	POLYGON
ARC	FILL	PRIVATE
ARRAY	FINAL	REAL
BASE	FONT	RIGHT
BASIC	HALF	SCALED
BIT	HATCH	SET
BODY	HEIGHT	SIZE
BOTTOM	HOLLOW	SOLID
BUNDLED	INDEX	START
CAP	INDEXED	STRING
CELL	INTEGER	STROKE
CHORD	LEFT	STYLE
CIRCLE	LINE	TABLE
CLIP	LIST	TEXT
CLOSE	MARKER	TRANSPARENCY
CODING	MESSAGE	TYPE
DATA	METRIC	UP
DEFAULTS	MODE	VALUE
DIRECT	NO	VDC
DOWN	NOT	VERSION
DRAWING	OFF	WIDTH
EDGE	ON	3 (numeral three)

ELLIPSE	PATH
EMPTY	PIE

5.4.4 Abbreviations

ABSOLUTE	ABS
ALIGNMENT	ALIGN
ALTERNATE	ALT
APPEND	APND
APPLICATION	APPL
ASPECT SOURCE FLAG(S)	ASF
AUXILIARY	AUX
BACKGROUND	BACK
BEGIN	BEG
CENTRE	CTR
CHARACTER	CHAR
COLOUR	COLR
CONTINUOUS	CONT
CONTROL	CTRL
DESCRIPTION	DESC
DISJOINT	DISJT
EIGHT	8
ELEMENT	ELEM
ELLIPTICAL	ELLIP
EXPANSION	EXPAN
EXTENDED	EXTD
GENERALIZED DRAWING PRIMITIVE	GDP
INCREMENTAL	INCR
INDIVIDUAL	INDIV
INTERIOR	INT
INVISIBLE	INVIS
MAXIMUM	MAX
METAFILE	MF
NORMAL	NORM
ORIENTATION	ORI
PATTERN	PAT
PICTURE	PIC
POINT	PT
POLYLINE	LINE
POLYMARKER	MARKER
PRECISION	PREC
RECTANGLE	RECT
REFERENCE	REF
RESTRICTED	RESTR
SCALING	SCALE
SEVEN	7
SPACING	SPACE
VISIBILITY	VIS
VISIBLE	VIS

5.4.5 The derived element names

Metafile Element	Element Name	Notes
BEGIN METAFILE	BEGMF	
END METAFILE	ENDMF	

Metafile format	Forming names	
BEGIN PICTURE	BEGPIC	
BEGIN PICTURE BODY	BEGPICBODY	
END PICTURE	ENDPIC	
METAFILE VERSION	MFVERSION	
METAFILE DESCRIPTION	MFDESC	
VDC TYPE	VDCTYPE	
INTEGER PRECISION	INTEGERPREC	
REAL PRECISION	REALPREC	
INDEX PRECISION	INDEXPREC	
COLOUR PRECISION	COLRPREC	
COLOUR INDEX PRECISION	COLRINDEXPREC	
MAXIMUM COLOUR INDEX	MAXCOLRINDEX	
COLOUR VALUE EXTENT	COLRVALUEEXT	
METAFILE ELEMENT LIST	MFELEMLIST	
METAFILE DEFAULTS REPLACEMENT	BEGMFDEFAULTS	(1)
	ENDMFDEFAULTS	
FONT LIST	FONTLIST	
CHARACTER SET LIST	CHARSETLIST	
CHARACTER CODING ANNOUNCER	CHARCODING	
SCALING MODE	SCALEMODE	
COLOUR SELECTION MODE	COLRMODE	
LINE WIDTH SPECIFICATION MODE	LINEWIDTHMODE	
MARKER SIZE SPECIFICATION MODE	MARKERSIZEMODE	
EDGE WIDTH SPECIFICATION MODE	EDGEWIDTHMODE	
VDC EXTENT	VDCEXT	
BACKGROUND COLOUR	BACKCOLR	
VDC INTEGER PRECISION	VDCINTEGERPREC	
VDC REAL PRECISION	VDCREALPREC	
AUXILIARY COLOUR	AUXCOLR	
TRANSPARENCY	TRANSPARENCY	
CLIP RECTANGLE	CLIPRECT	
CLIP INDICATOR	CLIP	
POLYLINE	LINE	(2)
	INCRLINE	
DISJOINT POLYLINE	DISJTLINE	(2)
	INCRDISJTLINE	
POLYMARKER	MARKER	(2)
	INCRMARKER	
TEXT	TEXT	
RESTRICTED TEXT	RESTRTEXT	
APPEND TEXT	APNDTEXT	
POLYGON	POLYGON	(2)
	INCRPOLYGON	
POLYGON SET	POLYGONSET	(2)
	INCRPOLYGONSET	
CELL ARRAY	CELLARRAY	
GENERALIZED DRAWING PRIMITIVE	GDP	
RECTANGLE	RECT	
CIRCLE	CIRCLE	
CIRCULAR ARC 3 POINT	ARC3PT	
CIRCULAR ARC 3 POINT CLOSE	ARC3PTCLOSE	
CIRCULAR ARC CENTRE	ARCCTR	
CIRCULAR ARC CENTRE CLOSE	ARCCTRCLOSE	
ELLIPSE	ELLIPSE	
ELLIPTICAL ARC	ELLIPARC	
ELLIPTICAL ARC CLOSE	ELLIPARCCLOSE	

Forming names	Metafile format
LINE BUNDLE INDEX	LINEINDEX
LINE TYPE	LINETYPE
LINE WIDTH	LINEWIDTH
LINE COLOUR	LINECOLR
MARKER BUNDLE INDEX	MARKERINDEX
MARKER TYPE	MARKERTYPE
MARKER SIZE	MARKERSIZE
MARKER COLOUR	MARKERCOLR
TEXT BUNDLE INDEX	TEXTINDEX
TEXT FONT INDEX	TEXTFONTINDEX
TEXT PRECISION	TEXTPREC
CHARACTER EXPANSION FACTOR	CHAREXPAN
CHARACTER SPACING	CHARSPACE
TEXT COLOUR	TEXTCOLR
CHARACTER HEIGHT	CHARHEIGHT
CHARACTER ORIENTATION	CHARORI
TEXT PATH	TEXTPATH
TEXT ALIGNMENT	TEXTALIGN
CHARACTER SET INDEX	CHARSETINDEX
ALTERNATE CHARACTER SET INDEX	ALTCHARSETINDEX
FILL BUNDLE INDEX	FILLINDEX
INTERIOR STYLE	INTSTYLE
FILL COLOUR	FILLCOLR
HATCH INDEX	HATCHINDEX
PATTERN INDEX	PATINDEX
EDGE BUNDLE INDEX	EDGEINDEX
EDGE TYPE	EDGETYPE
EDGE WIDTH	EDGEWIDTH
EDGE COLOUR	EDGECOLR
EDGE VISIBILITY	EDGEVIS
FILL REFERENCE POINT	FILLREFPT
PATTERN TABLE	PATTABLE
PATTERN SIZE	PATSIZE
COLOUR TABLE	COLRTABLE
ASPECT SOURCE FLAGS	ASF
ESCAPE	ESCAPE
MESSAGE	MESSAGE
APPLICATION DATA	APPLDATA

NOTES

1 This element is implemented by a pair of Clear Text element names in the metafile, one to begin defaults replacement and the second to end defaults replacement.

2 These elements have point list parameters, the points of which may be represented either with absolute or incremental coordinates. For each of these elements there are two possible Clear Text element names, one corresponding to absolute coordinate representation and one to incremental coordinate representation — the element name used shall correspond to the coordinate representation of point list.

6 Encoding the CGM elements

6.1 Allowable productions in clear text metafiles

All productions given below which do not appear in the table in 5.4.5 are merely "syntactic shorthand" for describing element productions, and may not appear by themselves in the metafile. For example, CELLROW is a handy way to describe a piece of the CELL ARRAY element, and is not a primitive in itself.

6.2 Encoding delimiter elements

BEGIN METAFILE ::= BEGMF
<OPTSEP>
<S:NAME>
<TERM>

END METAFILE ::= ENDMF <TERM>

BEGIN PICTURE ::= BEGPIC
<OPTSEP>
<S:PICTURENAME>
<TERM>

BEGIN PICTURE BODY ::= BEGPICBODY <TERM>

END PICTURE ::= ENDPIC <TERM>

6.3 Encoding metafile descriptor elements

METAFILE VERSION ::= MFVERSION
<SOFTSEP>
<I:VERSION>
<TERM>

METAFILE DESCRIPTION ::= MFDESC
<OPTSEP>
<S:DESCRIPTION>
<TERM>

VDC TYPE ::= VDCTYPE
<SOFTSEP>
< INTEGER | REAL >
<TERM>

INTEGER PRECISION ::= INTEGERPREC
<SOFTSEP>
<I:MININT>
<SEP>
<I:MAXINT>
<TERM>

The most negative and most positive integers (base 10) are given. These parameters are interpreted independently of all precisions currently set.

REAL PRECISION ::= REALPREC
<SOFTSEP>
<F:MINREAL>
<SEP>
<F:MAXREAL>
<SEP>
<I:DIGITS>
<TERM>

The parameters of this element are interpreted independently of all precisions currently set. The MINREAL and MAXREAL are signed real numbers giving the representable range of numbers. The DIGITS parameter gives the minimum number of DECIMAL DIGITS of accuracy assumed, and is of key importance in preventing roundoff error when the incremental forms of output primitives are used.

NOTE — This choice of format was patterned after the floating_point_constraint of the Ada programming language.

INDEX PRECISION ::= INDEXPREC
<SOFTSEP>
<I:MININT>
<SEP>
<I:MAXINT>
<TERM>

The most negative and most positive integers (base 10) are given. These parameters are interpreted independently of all precisions currently set.

COLOUR PRECISION ::= COLRPREC
<SOFTSEP>
<I:MAXCOMPONENT>
<TERM>

NOTE — Colour direct values are 3*I. COLOUR PRECISION gives a single integer range, 0..MAXCOMPONENT, within which each of the three compontents is contained. The parameters are interpreted independently of any currently set precisions.

COLOUR INDEX PRECISION ::= COLRINDEXPREC
<SOFTSEP>
<I:MAXINT>
<TERM>

The smallest colour index is 0. The most positive integer that may occur in a colour index parameter is given. This parameter is interpreted independently of all precisions currently set. See MAXCOLRINDEX.

MAXIMUM COLOUR INDEX ::= MAXCOLRINDEX
<SOFTSEP>
<I:MAXINDEXVALUE>
<TERM>

The MAXINDEXVALUE shall be less than or equal to the <MAXINT> parameter of COLRINDEXPREC.

```
COLOUR VALUE EXTENT      ::= COLRVALUEEXT
                             <SOFTSEP>
                             <BLACK:RED GREEN BLUE>
                             <SEP>
                             <WHITE:RED GREEN BLUE>
                             <TERM>

METAFILE ELEMENT LIST    ::= MFELEMLIST
                             <OPTSEP>
                             <S:ELEMENTNAMES>
                             <TERM>
```

The string parameter consists of a list of element names separated by <SEP>. In addition, the words DRAWINGPLUS and DRAWINGSET may be used in this string, as defined in ISO 8632/1.

```
METAFILE DEFAULTS REPLACEMENT ::= BEGMFDEFAULTS
                                  <TERM>

                                     <ELEMENT>+

                                  ENDMFDEFAULTS
                                  <TERM>
```

Between the two bracketing elements, applicable CGM elements appear using the same format as described elsewhere in this section.

```
FONT LIST                ::= FONTLIST
                             <OPTSEP>
                             <S:FONTNAME>
                             < <SEP> <S:FONTNAME> >*
                             <TERM>

CHARACTER SET LIST       ::= CHARSETLIST
                             <CHARSETDESIGNATOR>+
                             <TERM>

CHARSETDESIGNATOR        ::= <SOFTSEP>
                             < STD94 |
                               STD96 |
                               STD94MULTIBYTE |
                               STD96MULTIBYTE |
                               COMPLETECODE >
                             < <OPTSEP> | <SEP> >
                             <S:TAIL>


CHARACTER CODING ANNOUNCER   ::= CHARCODING
                                 < BASIC7BIT |
                                   BASIC8BIT |
                                   EXTD7BIT |
                                   EXTD8BIT >
                                 <TERM>
```

6.4 Encoding picture descriptor elements

SCALING MODE ::= SCALEMODE
<SOFTSEP>
< ABSTRACT | METRIC >
<SEP>
<R:SCALEFACTOR>
<TERM>

COLOUR SELECTION MODE ::= COLRMODE
<SOFTSEP>
< INDEXED | DIRECT >
<TERM>

LINE WIDTH SPECIFICATION MODE ::= LINEWIDTHMODE
<SOFTSEP>
< ABSTRACT | SCALED >
<TERM>

MARKER SIZE SPECIFICATION MODE ::= MARKERSIZEMODE
<SOFTSEP>
< ABSTRACT | SCALED >
<TERM>

EDGE WIDTH SPECIFICATION MODE ::= EDGEWIDTHMODE
<SOFTSEP>
< ABSTRACT | SCALED >
<TERM>

VDC EXTENT ::= VDCEXT
<SOFTSEP>
<P:FIRSTCORNER>
<SEP>
<P:SECONDCORNER>
<TERM>

BACKGROUND COLOUR ::= BACKCOLR
<SOFTSEP>
<RED GREEN BLUE>
<TERM>

6.5 Encoding control elements

VDC INTEGER PREC ::= VDCINTEGERPREC
<SOFTSEP>
<I:MININT>
<SEP>
<I:MAXINT>
<TERM>

See INTEGERPREC for description of the parameters.

VDC REAL PRECISION ::= VDCREALPREC
<SOFTSEP>

<F:MINREAL>
<SEP>
<F:MAXREAL>
<SEP>
<I:DIGITS>
<TERM>

See REALPREC for description of the parameters.

AUXILIARY COLOUR ::= AUXCOLR
<SOFTSEP>
<K:AUXCOLR>
<TERM>

TRANSPARENCY ::= TRANSPARENCY
<SOFTSEP>
< OFF ¦ ON >
<TERM>

CLIP RECTANGLE ::= CLIPRECT
<SOFTSEP>
<P:FIRSTCORNER>
<SEP>
<P:SECONDCORNER>
<TERM>

CLIP INDICATOR ::= CLIP
<SOFTSEP>
< OFF ¦ ON >
<TERM>

6.6 Encoding graphical primitive elements

POLYLINE ::= < LINE
<POINTLIST>
<TERM>
>
¦
< INCRLINE
<INCRPOINTLIST>
<TERM>
>

LINE and INCRLINE shall always have at least two points.

POINTLIST ::= <SOFTSEP>
<P:POINT>
< <SEP> <P:POINT> >+

INCRPOINTLIST ::= <SOFTSEP>
<P:FIRSTPOINT>
<DELTA>+

DELTA ::= < <SEP> <DELTAPAIR> > ¦
< <SEP> <LEFT PAREN> <DELTAPAIR> <RIGHT PAREN> >

```
DELTAPAIR           ::= <OPTSEP>
                        <VDC:DELTAX>
                        <SEP>
                        <VDC:DELTAY>
                        <OPTSEP>
```

NOTE — Absolute coordinates correspond in a straightforward way to the functionality definition, but incremental coordinates can realize a significant savings in this encoding. Even more important, incremental coordinates are high on the priority list for user-friendly graphics, which is a high priority objective of this encoding.

```
DISJOINT POLYLINE   ::= < DISJTLINE
                          <SOFTSEP>
                          <P:POINT>
                          <SEP>
                          <P:POINT> >
                          < <SEP>
                            <P:POINT>
                            <SEP>
                            <P:POINT> >*
                          <TERM>
                        >
                        |
                        < INCRDISJTLINE
                          <SOFTSEP>
                          <P:POINT>
                          <DELTA>
                          < <DELTA> <DELTA> >*
                          <TERM>
                        >
```

DISJTLINE and INCRDISTJTLINE shall have pairs of points, and at least one pair.

```
POLYMARKER          ::= < MARKER
                          <POINTLIST>
                          <TERM>
                        >
                        |
                        < INCRMARKER
                          <INCRPOINTLIST>
                          <TERM>
                        >

TEXT                ::= TEXT
                          <SOFTSEP>
                          <P:TEXTLOCATION>
                          <SEP>
                          <TEXTPIECE>

RESTRICTED TEXT     ::= RESTRTEXT
                          <SOFTSEP>
                          <VDC:MAXWIDTH>
                          <SEP>
                          <VDC:MAXHEIGHT>
                          <SEP>
```

```
                        <P:TEXTLOCATION>
                        <SEP>
                        <TEXTPIECE>

APPEND TEXT         ::= APNDTEXT
                        <SOFTSEP>
                        <TEXTPIECE>

TEXTPIECE           ::= < NOTFINAL ¦ FINAL >
                        < <OPTSEP> ¦ <HARDSEP> >
                        <S:TEXTSTRING>
                        <TERM>

POLYGON             ::=  < POLYGON
                           <SOFTSEP>
                           <P:POINT>
                           <SEP>
                           <P:POINT>
                           < <SEP> <P:POINT> >+
                           <TERM>
                        >
                        ¦
                        < INCRPOLYGON
                           <SOFTSEP>
                           <P:POINT>
                           <DELTA>
                           <DELTA>+
                           <TERM>
                        >

POLYGON SET         ::= < POLYGONSET
                           <SOFTSEP>
                           <FLAGGEDPOINT>
                           <SEP>
                           <FLAGGEDPOINT>
                           < <SEP> <FLAGGEDPOINT> >+
                           <TERM>
                        >
                        ¦
                        < INCRPOLYGONSET
                           <SOFTSEP>
                           <FLAGGEDPOINT>
                           < <FLAGGEDDELTA> <SEP> >+
                           <FLAGGEDDELTA>
                           <TERM>
                        >
```

For all POLYGON-type primitives, at least three points shall be present.

```
EDGEFLAG            ::= INVIS ¦ VIS ¦ CLOSEINVIS ¦ CLOSEVIS

FLAGGEDPOINT        ::= <P:VERTEX> <SEP> <EDGEFLAG>

FLAGGEDDELTA        ::= <DELTA> <SEP> <EDGEFLAG>

CELL ARRAY          ::= CELLARRAY
```

<SOFTSEP>
<P:P_POINT>
<SEP>
<P:Q_POINT>
<SEP>
<P:R_POINT>
<SEP>
<I:NX>
<SEP>
<I:NY>
<LOCLCOLRPREC>
<CELLROW>*
<TERM>

LOCLCOLRPREC ::= < <SEP> <I:MAXINT> > ¦
< <SEP> <I:MAXCOMPONENT> > ¦
< <SEP> <I:0> >

The LOCLCOLRPREC takes the form of COLRINDEXPREC or COLRPREC, depending on whether the COLRMODE is <INDEXED> or <DIRECT>, respectively. The value 0 is the 'default precision indicator', and denotes that the precision currently in effect shall be used.

CELLROW ::= < <SEP> <CELLLIST> > ¦
< <SEP> <LEFT PAREN>
<CELLLIST> <RIGHT PAREN> >

CELLLIST ::= <NULL> ¦
< <K:CELL> < <SEP> <K:CELL> >* >

The parenthesized form of the list is optional. If parentheses are used, they delimit a row of cells. Each row is considered to start from the side defined by the points (P,Q,R) as defined in ISO 8632/1; if a row is not complete, it is left as a metafile-interpreter dependency whether the rest of the row is left untouched or whether the last cell in the row is replicated to fill the row. If there are too many cells in a row, the surplus cells are thrown away. The parentheses may alternately be treated as SPACE characters, in which case the cells are processed sequentially and no row adjusting is done.

The local colour precision parameter, LOCLCOLRPREC, takes the form of the parameter of a colour-precision-setting element, either of COLRPREC or of COLRINDEXPREC, depending on the COLRMODE. Legal values are either legal values of one of those colour precision elements, or zero. If the value is zero, then the metafile's COLRPREC or COLRINDEXPREC is to be used within the CELLARRAY element as well.

GENERALIZED DRAWING PRIMITIVE ::= GDP
<SOFTSEP>
<I:GDP_IDENTIFIER>
<SEP>
<POINTLIST>
< <OPTSEP> ¦ <HARDSEP> >
<S:DATARECORD>
<TERM>

RECTANGLE ::= RECT
<SOFTSEP>
<P:FIRSTCORNER>

<SEP>
<P:SECONDCORNER>
<TERM>

CIRCLE ::= CIRCLE
<SOFTSEP>
<P:CENTRE>
<SEP>
<VDC:RADIUS> {non-negative}
<TERM>

CIRCULAR ARC 3 POINT ::= ARC3PT
<3PTARCSPEC>
<TERM>

CIRCULAR ARC 3 POINT CLOSE ::= ARC3PTCLOSE
<3PTARCSPEC>
<CLOSESPEC>
<TERM>

CLOSESPEC ::= <SEP>
< PIE ¦ CHORD >

3PTARCSPEC ::= <SOFTSEP>
<P:STARTPOINT>
<SEP>
<P:INTERMEDIATEPOINT>
<SEP>
<P:ENDPOINT>

CIRCULAR ARC CENTRE ::= ARCCTR
<CTRARCSPEC>
<TERM>

CIRCULAR ARC CENTRE CLOSE ::= ARCCTRCLOSE
<CTRARCSPEC>
<CLOSESPEC>
<TERM>

CTRARCSPEC ::= <SOFTSEP>
<P:CENTREPOINT>
<ARCBOUNDS>
<SEP>
<VDC:RADIUS> {non-negative}

ARCBOUNDS ::= <SEP>
<P:STARTVECTOR>
<SEP>
<P:ENDVECTOR>

NOTE — The start and end vectors are given as point records rather than 2*VDC to permit the parenthesized form to be used to represent the vectors.

ELLIPSE ::= ELLIPSE
<ELLIPSESPEC>
<TERM>

ELLIPSESPEC ::= <SOFTSEP>
<P:CENTRE>
<SEP>
<P:ENDPOINT_FIRST_CONJUGATE_DIAMETER>
<SEP>
<P:ENDPOINT_SECOND_CONJUGATE_DIAMETER>

ELLIPTICAL ARC ::= ELLIPARC
<ELLIPSESPEC>
<ARCBOUNDS>
<TERM>

ELLIPTICAL ARC CLOSE ::= ELLIPARCCLOSE
<ELLIPSESPEC>
<ARCBOUNDS>
<CLOSESPEC>
<TERM>

6.7 Encoding attribute elements

LINE BUNDLE INDEX ::= LINEINDEX
<SOFTSEP>
<I:BUNDLEINDEX> {positive}
<TERM>

LINE TYPE ::= LINETYPE
<SOFTSEP>
<I:LINETYPE>
{ 1=solid, 2=dash, 3=dot,
4=dash-dot, 5=dash-dot-dot,
<0 implementation dependent}
<TERM>

LINE WIDTH ::= LINEWIDTH
<SOFTSEP>
<V:LINEWIDTH> {non-negative}
<TERM>

LINE COLOUR ::= LINECOLR
<SOFTSEP>
<K:LINECOLR>
<TERM>

MARKER BUNDLE INDEX ::= MARKERINDEX
<SOFTSEP>
<I:BUNDLEINDEX> {positive}
<TERM>

MARKER TYPE ::= MARKERTYPE
<SOFTSEP>
<I:MARKERTYPE>
{ 1=dot, 2=plus, 3=asterisk, 4=circle
5=cross (x), <0 implementation dependent }
<TERM>

```
MARKER SIZE                    ::= MARKERSIZE
                                     <SOFTSEP>
                                     <V:MARKERSIZE>  {non-negative}
                                   <TERM>

MARKER COLOUR                  ::= MARKERCOLR
                                     <SOFTSEP>
                                     <K:MARKERCOLR>
                                   <TERM>

TEXT BUNDLE INDEX              ::= TEXTINDEX
                                     <SOFTSEP>
                                     <I:BUNDLEINDEX>  {positive}
                                   <TERM>

TEXT FONT INDEX                ::= TEXTFONTINDEX
                                     <SOFTSEP>
                                     <I:FONTINDEX>    {positive}
                                   <TERM>

TEXT PRECISION                 ::= TEXTPREC
                                     <SOFTSEP>
                                     < STRING ¦ CHAR ¦ STROKE >
                                   <TERM>

CHARACTER EXPANSION FACTOR     ::= CHAREXPAN
                                     <SOFTSEP>
                                     <R:FACTOR>
                                   <TERM>

CHARACTER SPACING              ::= CHARSPACE
                                     <SOFTSEP>
                                     <R:SPACING>
                                   <TERM>

TEXT COLOUR                    ::= TEXTCOLR
                                     <SOFTSEP>
                                     <K:TEXTCOLR>
                                   <TERM>

CHARACTER HEIGHT               ::= CHARHEIGHT
                                     <SOFTSEP>
                                     <VDC:CHARHEIGHT>  {non-negative}
                                   <TERM>

CHARACTER ORIENTATION          ::= CHARORI
                                     <SOFTSEP>
                                     <DELTAPAIR> {up vector}
                                     <SEP>
                                     <DELTAPAIR> {base vector}
                                   <TERM>

TEXT PATH                      ::= TEXTPATH
                                     <SOFTSEP>
                                     < RIGHT ¦ LEFT ¦ UP ¦ DOWN >
                                   <TERM>
```

```
TEXT ALIGNMENT          ::= TEXTALIGN
                              <SOFTSEP>
                              < NORMHORIZ ¦ LEFT ¦ CTR ¦ RIGHT ¦ CONTHORIZ >
                              <SEP>
                              < NORMVERT ¦ TOP ¦ CAP ¦ HALF ¦ BASE ¦ BOTTOM ¦
                                CONTVERT >
                              <SEP>
                              <R:CONTINUOUS_HORIZONTAL>
                              <SEP>
                              <R:CONTINUOUS_VERTICAL)
                            <TERM>

CHARACTER SET INDEX     ::= CHARSETINDEX
                              <SOFTSEP>
                              <I:CHARSETINDEX> {positive}
                            <TERM>

ALTERNATE CHARACTER SET INDEX::=ALTCHARSETINDEX
                              <SOFTSEP>
                              <I:ALTCHARSETINDEX> {positive}
                            <TERM>

FILL BUNDLE INDEX       ::= FILLINDEX
                              <SOFTSEP>
                              <I:BUNDLEINDEX> {positive}
                            <TERM>

INTERIOR STYLE          ::= INTSTYLE
                              <SOFTSEP>
                              < HOLLOW ¦ SOLID ¦ PAT ¦ HATCH ¦ EMPTY >
                            <TERM>

FILL COLOUR             ::= FILLCOLR
                              <SOFTSEP>
                              <K:FILLCOLR>
                            <TERM>

HATCH INDEX             ::= HATCHINDEX
                              <SOFTSEP>
                              <I:HATCHINDEX>
                                { 1=horizontal,
                                  2=vertical
                                  3=positive slope
                                  4=negative slope
                                  5=horiz/vertical cross
                                  6= +/- slope cross
                                  <0 implementation dependent}
                              <TERM>

PATTERN INDEX           ::= PATINDEX
                              <SOFTSEP>
                              <I:PATINDEX> {positive}
                            <TERM>

EDGE BUNDLE INDEX       ::= EDGEINDEX
                              <SOFTSEP>
```

<I:BUNDLEINDEX> {positive}
<TERM>

EDGE TYPE ::= EDGETYPE
<SOFTSEP>
<I:EDGETYPE>
{ 1=solid, 2=dash, 3=dot,
4=dash-dot, 5=dash-dot-dot,
<0 implementation dependent }
<TERM>

EDGE WIDTH ::= EDGEWIDTH
<SOFTSEP>
<V:EDGEWIDTH> {non-negative}
<TERM>

EDGE COLOUR ::= EDGECOLR
<SOFTSEP>
<K:EDGECOLR>
<TERM>

EDGE VISIBILITY ::= EDGEVIS
<SOFTSEP>
< OFF | ON >
<TERM>

FILL REFERENCE POINT ::= FILLREFPT
<SOFTSEP>
<P:FILLREFPT>
<TERM>

PATTERN TABLE ::= PATTABLE
<SOFTSEP>
<I:PATINDEX>
<SEP>
<I:NX>
<SEP>
<I:NY>
<LOCLCOLRPREC>
<CELLROW>*
<TERM>

See CELL ARRAY for discussion of the LOCLCOLRPREC and CELLROW productions.

PATTERN SIZE ::= PATSIZE
<SOFTSEP>
<DELTAPAIR> {height vector}
<SEP>
<DELTAPAIR> {width vector}
<TERM>

COLOUR TABLE ::= COLRTABLE
<SOFTSEP>
<I:STARTINGINDEX> {non-negative}
< <SEP> <RED GREEN BLUE> >*
<TERM>

```
ASPECT SOURCE FLAGS   ::= ASF
                           <SOFTSEP>
                           <ASFPAIR>
                           < <SEP> <ASFPAIR> >*
                         <TERM>

ASFPAIR               ::= < ASFTYPE >
                           <SEP>
                           < INDIV ¦ BUNDLED >

ASFTYPE               ::= < ASFNAME ¦ PSEUDOASF >

ASFNAME               ::= < LINETYPE ¦ LINEWIDTH ¦ LINECOLR ¦
                            MARKERTYPE ¦ MARKERSIZE ¦ MARKERCOLR ¦
                            TEXTFONTINDEX ¦ TEXTPREC ¦ CHAREXP ¦
                            CHARSPACE ¦ TEXTCOLR ¦ INTSTYLE ¦
                            FILLCOLR ¦ HATCHINDEX ¦ PATINDEX ¦
                            EDGETYPE ¦ EDGEWIDTH ¦ EDGECOLR >

PSEUDOASF             ::= < ALL ¦ ALLLINE ¦ ALLMARKER ¦ ALLTEXT ¦
                            ALLFILL ¦ ALLEDGE >
```

The ASF type may either be a valid ASF name, or a pseudo-ASF. If the former, then the name shall match the name of the corresponding attribute element.

NOTE — The pseudo-ASFs are a shorthand convenience for setting a number of ASFs at once.

The pseudo-ASFs have the meanings:

ALL: set all ASFs as indicated.
ALLLINE: set LINETYPE, LINEWIDTH, and LINECOLR ASFs as indicated.
ALLMARKER: set MARKERTYPE, MARKERSIZE, and MARKERCOLR ASFs as indicated.
ALLTEXT: set TEXTFONTINDEX, TEXTPREC, CHAREXP, CHARSPACE, and TEXTCOLR ASFs as indicated.
ALLFILL: set INTSTYLE, FILLCOLR, HATCHINDEX, and PATINDEX ASFs as indicated.
ALLEDGE: set EDGETYPE, EDGEWIDTH, and EDGECOLR as indicated.

6.8 Encoding escape elements

```
ESCAPE                ::= ESCAPE
                           <SOFTSEP>
                           <I:IDENTIFIER>
                           < <OPTSEP> ¦ <HARDSEP> >
                           <S:DATARECORD>
                         <TERM>
```

6.9 Encoding external elements

```
MESSAGE               ::= MESSAGE
                           <SOFTSEP>
                           < NOACTION ¦ ACTION >
                           < <OPTSEP> ¦ <HARDSEP> >
                           <S:MESSAGE_TEXT>
                         <TERM>
```