POSTSCRIPT®

ADOBE ILLUSTRATOR® DOCUMENT FORMAT Specification Version 2.0

July 18, 1990
PostScript® Developer Support Group

Adobe Systems Incorporated
1585 Charleston Road PO Box 7900
Mountain View, CA 94039-7900
(415) 961-4400

PN LPS5007

POSTSCRIPT®

ADOBE ILLUSTRATOR® DOCUMENT FORMAT Specification Version 2.0

July 18, 1990
PostScript® Developer Support Group
(415) 961-4111

1. INTRODUCTION

This Technical Note describes the format of the document files in which *Adobe Illustrator®* stores graphic illustrations. An *Adobe Illustrator* document is a PostScript® page description which conforms to the Adobe PostScript *Document Structuring Conventions*, version 2.0. The page description may be executed by any PostScript interpreter to render the illustration on a display or on a printed page. The document may also be imported by page composition systems and other illustration editing applications.

Adobe Illustrator can store a document in *Encapsulated PostScript File* (EPS) format. A low resolution (72 dpi) image of the illustration may be included as part of the EPS file as an option. This image may be used by a page composition system for picture placement, scaling, cropping, and previewing.

Although an intimate knowledge of the Postscript page description language is not essential for understanding this document there are several notions that the reader should understand. The principal notion is how the desired marks on a page are described. PostScript page descriptions are based on the notion of a *path*. A path is a mathematical line constructed from possibly disjoint segments. Path segments may be either straight lines or *Bézier* curves. A path may be open, meaning that the beginning and end points are distinct, or closed, meaning that the beginning and end points on the path are the same. Opaque ink is put on the page either by *stroking* the path to trace it with a line of a given thickness or by *filling* the path to put ink everywhere inside the path. The thickness of the line used to stroke a path, the color of the ink to use, and so on, are attributes of the PostScript *graphics state* which may be changed by a PostScript program to control how paths are marked. Text is a special case of the more general drawing facilities in the language. Individual characters are rendered by pre-defined PostScript programs which construct and then fill or stroke a path. The second important notion is that the execution model for the language is based on stacks. The operands for an operator are pushed onto a stack and then the operator is executed. Each operator pops its operands off the stack and pushes its results back onto the stack.

The coordinate system used by Adobe Illustrator is the same as the PostScript language *default user space* in which the origin is in the lower left-hand corner of the page, the *x* axis extends horizontally to the right, and the *y* axis extends vertically upward. The length of a unit along each axis is 1/72 of an inch, which is approximately one printer's *point* (1/72.27 inch).

The reader is referred to the *PostScript Language Reference Manual* (Addison-Wesley, ISBN 0-201-10174-2), *Document Structuring Conventions*, version 2.0 (Technical Note LPS5001), and *Encapsulated PostScript Files*, version 1.2 (Technical Note LPS5002) for a full specification of the PostScript page description language and the conventions for structuring PostScript documents. The reader is assumed to be familiar with the terms used in the *Adobe Illustrator 88 User Guide*.

2. DOCUMENT OVERVIEW

Adobe Illustrator uses a small, specialized language to describe the graphic elements of an illustration. The special language is much simpler than the full PostScript language. In particular, the language is purely declarative in that there are no control constructs for performing iterative or conditional computations. Each of the operators in the language can be emulated by a PostScript procedure. An Adobe Illustrator document contains both the PostScript language definitions for the operators that constitute the special illustration language and the illustration itself, which is described in the illustration language. The definitions and illustration are stored as a structured PostScript document description.

A PostScript document description that obeys the *Document Structuring Conventions* has two main parts: A prologue and a script. The script has three parts: a setup sequence, a sequence of page descriptions, and a trailer. A *conforming* document is one that obeys a proper subset of the structuring conventions. Specifically there must not be any PostScript code executed in the prologue and no definitions made in the script. Adobe Illustrator creates conforming structured PostScript files.

The *prologue* part of a document encapsulates information needed both by other programs that need to interpret the file and PostScript language definitions of procedures and variables that are used in the individual page descriptions. For Adobe Illustrator, the prologue defines the collection of PostScript procedures that implement the operators of the special illustration language. The main body of the document is a *script* which describes the illustration using the illustration language.

The document structuring information is embedded in a PostScript program using stylized comments. Apart from the first comment in a file, the structuring comments all have the form:

%%Keyword{: *arguments*}

Many structuring comments require information in addition to the keyword. This information is separated from the keyword by a colon and continues to the newline character that terminates the comment.

The syntax for describing the Adobe Illustrator document format here is described using a common notation, known as BNF (*Backus-Naur Form*):

```
  <xyz> ::=              abc <def> ghi |
                         <k> j
```

A token enclosed in angle brackets names a class of document component, while plain text appears verbatim or with some obvious substitution. The grammar rules have two parts. On the left of the “::=” definition symbol is the name of a class of components. In the example above, the class is named *xyz*. On the right of the definition symbol is a set of one or more alternative forms that an *xyz* component might be take in the document. The alternative forms are separated by the vertical bar character (|). Each line on the right hand side of the rule corresponds to one line in the document. If the only content of a line on the right hand side is another class name, then the line may represent more than one line in the document. Most of this will become clear as the individual document components are described. Single letter components, such as “<A>”, refer to the corresponding illustration language operator **A**. The notation “{ ... }” means that the items enclosed in curly brackets are optional. If the curly brackets are followed by an asterisk (*), the objects inside the brackets may be repeated zero or more times.

Using the notation described above, the overall structure of an Adobe Illustrator document is:

```
<document> ::=        <prologue>
                      <script>

<script> ::=          <setup>
                      <script body>
                      <trailer>
```

The full document syntax structure is summarized in Section 7.

The syntax and semantics of the individual operators of the illustration language are defined in later sections of this document. Each operator definition looks like the following:

operand$_1$*... operand*$_m$ **op**

The functionality of the **op** *operator is described in here.*

This notation means that the operator **op** takes *m* operands and performs some operation. Each operand is characterized either by its data type (e.g., *integer*) or a more meaningful name (e.g., *linewidth*). In the latter case, the range of legitimate values is given in the description. A dash (–) is used to indicate that an operator requires no operands.

There are several versions of *Adobe Illustrator* available: *Adobe Illustrator*, *Adobe Illustrator — Windows Version*, *Adobe Illustrator 88*, and *Adobe Illustrator Japanese Edition*. Differences in the document formats among the various versions will be indicated in the descriptions of the individual components of the document. Parts of the document or individual operators that are used exclusively by Adobe Illustrator 88 are marked by “**[AI88]**.” Where distinction between Adobe Illustrator 88 and other versions of Adobe Illustrator are mentioned in this document, Adobe Illustrator Japanese Edition is considered equivalent to Adobe Illustrator 88 unless specifically noted otherwise.

Sections 3 to 7 describe the format of document files written by Adobe Illustrator. Section 8 describes how standard information can be removed from documents written by Adobe Illustrator to minimize their size for transmission and storage. Section 9 illustrates how minimal documents may be created for later editing with Adobe Illustrator. Section 10 describes the contents of the resource fork that is part of an Adobe Illustrator document on the Macintosh®.

3. PROLOGUE

The syntax for an Adobe Illustrator document prologue is:

```
<prologue> ::=          %!PS-Adobe-2.0 EPSF-1.2
                        <prologue body>
                        %%EndProlog
```

%!PS-Adobe-2.0 EPSF-1.2

The first line of the file declares with the "%!" comment prefix that the file contains a PostScript program. It then specifies that the file conforms to version 2.0 of the *Document Structuring Conventions* and further that the file conforms to version 1.2 of the conventions for *Encapsulated PostScript Files*. Both of the version numbers must correspond to the specification used for constructing the document.

%%EndProlog

The **%%EndProlog** comment marks the end of the entire document prologue.

The prologue body contains some header information and the definitions of any PostScript procedures used in the document. The syntax for the prologue body is:

```
<prologue body> ::=     <header>
                        <proc sets>
```

3.1 Header

The header for the prologue of an Adobe Illustrator document follows the "**%!**" comment on the first line of the file. The syntax for the prologue header is:

```
<header> ::=            <header comments>
                        %%EndComments
```

%%EndComments

The **%%EndComments** comment marks the end of the header part of the prologue.

The sequence of header comments is a subset of the list below. The syntax for the individual comments is described informally. Almost all of the header comments are optional. The comments required by Adobe Illustrator in all documents are marked "**[Required]**." Some comments are required only if a specific feature is used in an illustration. These comments are marked "**[As Necessary]**."

Some of the comments used in the header are not defined in the official *Document Structuring Conventions*, but are useful to applications which need to handle Adobe Illustrator documents. These comments are identified explicitly.

```
%%Creator: Adobe Illustrator 88(TM) version
%%For: (username) (organization)
%%Title: (Illustration title)
%%CreationDate: (date) (time)
%%DocumentProcSets: Adobe_packedarray level revision
%%DocumentSuppliedProcSets: Adobe_packedarray level revision
%%DocumentProcSets: Adobe_cmykcolor level revision
%%DocumentSuppliedProcSets: Adobe_cmykcolor level revision
%%DocumentProcSets: Adobe_cshow level revision
%%DocumentSuppliedProcSets: Adobe_cshow level revision
%%DocumentProcSets: Adobe_customcolor level revision
%%DocumentSuppliedProcSets: Adobe_customcolor level revision
%%DocumentProcSets: Adobe_pattern level revision
%%DocumentSuppliedProcSets: Adobe_pattern level revision
%%DocumentProcSets: Adobe_Illustrator88version level revision
%%DocumentSuppliedProcSets: Adobe_Illustrator88version level revision
%%DocumentFonts: font ...
%%+font ...
%%DocumentFiles: filename ...
%%+filename ...
%%ColorUsage: keyword
%%DocumentProcessColors: keyword ...
%%DocumentCustomColors: (customcolorname)
%%+ (customcolorname) ...
%%CMYKCustomColor: cyan magenta yellow black (customcolorname)
%%+ cyan magenta yellow black (customcolorname)
%%BoundingBox:llx lly urx ury
%%TemplateBox:llx lly urx ury
%%TileBox:llx lly urx ury
```

The individual lines in the header are specified as follows:

%%Creator: Adobe Illustrator 88(TM) *version*

The **%%Creator** comment identifies the application that generated the PostScript document. The *version* number is arbitrary text terminated by a newline character.

%%For: (*username*) (*organization*)

The **%%For** comment identifies the user who created the file and the organization to which the user belongs. Both the *username* and *organization* are valid PostScript language strings. The PostScript language string escape sequences for including characters outside the printable ASCII character set and for representing the "(", ")", and other special characters in strings are discussed in Section 3.3 of the *PostScript Language Reference Manual*.

%%Title: (*Illustration title*)

The **%%Title** comment provides an arbitrary text title for the document. The title is a valid PostScript language string as above.

%%CreationDate: (*date*) (*time*)

The **%%CreationDate** comment gives the date and time that the document was created. The *date* and *time* are each valid PostScript language strings as above.

%%DocumentProcSets: Adobe_Illustrator88 *level revision*

The **%%DocumentProcSets** comment specifies that a procedure set named **Adobe_Illustrator88** is required by the illustration and specifically that the version identified by *level* and *revision* of that procedure set is required. The *level* and *revision* are arbitrary text delimited by whitespace.

%%DocumentSuppliedProcSets: Adobe_Illustrator88 *level revision*

The **%%DocumentSuppliedProcSets** comment specifies that the procedure set named **Adobe_Illustrator88** is defined later in the prologue (see following section).

Note*: Adobe Illustrator uses two versions of its main procedure set. The procedure set named* Adobe_Illustrator88 *is the full version. However, if no patterns are used in an Adobe Illustrator 88 document, the* Adobe_Illustrator881 *procedure set is used instead, saving over 4K bytes in the size of the document file. (See Section 8,* ***Compressing Illustrator Documents****.)*

%%DocumentProcSets: Adobe_packedarray *level revision*
%%DocumentSuppliedProcSets: Adobe_packedarray *level revision*
%%DocumentProcSets: Adobe_cmykcolor *level revision*
%%DocumentSuppliedProcSets: Adobe_cmykcolor *level revision*
%%DocumentProcSets: Adobe_cshow *level revision*
%%DocumentSuppliedProcSets: Adobe_cshow *level revision*
%%DocumentProcSets: Adobe_customcolor *level revision*
%%DocumentSuppliedProcSets: Adobe_customcolor *level revision*
%%DocumentProcSets: Adobe_pattern *level revision*
%%DocumentSuppliedProcSets: Adobe_pattern *level revision*

These comments indicate that the named procedure sets are both required and defined by the document (see following section). Comments appear only for the actual procedure sets needed by the illustration. (See Section 8, **Compressing Illustrator Documents**.) **[AI88]**

%%DocumentFonts: *font ...*
%%+*font ...*

The **%%DocumentFonts** comment enumerates the names of PostScript fonts that are used in the document. The **%%+** comment indicates a continuation of the list of keywords from the previous comment. In this case it extends the list of fonts used in the document. This list includes fonts which are listed in the **%%DocumentFonts** comment in any files which are included (placed) within an Adobe Illustrator 88 document (see section 5.8). The fonts may need to be downloaded to the PostScript printer before the document description can be executed. This comment should be omitted if no fonts are used in the document.

%%DocumentFiles: *filename ...*
%%+*filename ...*

The **%%DocumentFiles** comment names files that need to be imported to complete the illustration. The site at which the file is needed is marked by another comment, **%%IncludeFile**. See the Section 5.8, **Imported Document Operators** below. This comment should be omitted if no files are imported into the document. **[AI88]**

%%ColorUsage: *keyword*

The **%%ColorUsage** comment indicates whether the document uses only black or colored ink, indicated by the keywords **Black&White** and **Color**, respectively. *This is not an official structuring comment.* **[AI88]**

%%DocumentProcessColors: *keyword ...*

The **%%DocumentProcessColors** comment specifies which of the process colors identified by the keywords **Cyan**, **Magenta**, **Yellow**, and **Black** are used in the document. This comment is used by programs producing color separations. **[AI88]**

%%DocumentCustomColors: (*customcolorname*)
%%+ (*customcolorname*)

The **%%DocumentCustomColors** comment enumerates the names of the custom colors used in the document. The names are valid PostScript strings which are enclosed in parentheses. For example the PANTONE® colors are identified by names such as: "(PANTONE 156 CV)". The list of custom color names may be continued on subsequent lines, each of which must begin with the "**%%+**" prefix. **[AI88**, **As Necessary]**

%%CMYKCustomColor: *cyan magenta yellow black* (*customcolorname*)
%%+ *cyan magenta yellow black* (*customcolorname*)

The **%%CMYKCustomColor** comment specifies an *approximation* for the named custom color in terms of the four components: cyan, magenta, yellow, and black. Each component value must be a real value in the range [0.0,1.0]. The component values are analogous to the arguments to the PostScript **setcmykcolor** operator. **[AI88**, **As Necessary]**

%%BoundingBox:*llx lly urx ury*

The **%%BoundingBox** comment specifies the imaginary box that encloses all of the marks painted on the page for the illustration. The integer coordinates are specified in the default user coordinate system. *Some versions of Adobe Illustrator* incorrectly *used real numbers instead of integers for the bounding box.* The width (*urx* - *llx*) and height (*ury* - *lly*) of the bounding box must be integers. **[Required]**

%%TemplateBox:*llx lly urx ury*

The **%%TemplateBox** comment specifies the imaginary box that encloses all of the samples in the document's template (see *Adobe Illustrator 88 User Guide*). The integer (or real) coordinates are specified in the default user coordinate system. Each sample in the template is assumed to be 1/72 inch square. The width (*urx* - *llx*) and height (*ury* - *lly*) of the template box must be integers. If the document has no template, then the width and height of the template bounding box must be zero. *This is not an official structuring comment.* **[Required]**

When Adobe Illustrator opens a document, the illustration is placed on the drawing area in such a way that the coordinate (llx + urx)/2, (lly + ury)/2) is centered in the drawing area.

%%TileBox:*llx lly urx ury*

The **%%TileBox** comment specifies the bounding box of the main tile of the drawing area (see *Adobe Illustrator 88 User Guide*). The initial ruler position is centered in this box. *This is not an official structuring comment.*

The subset of the *Adobe Illustrator 88* header described above used by *Adobe Illustrator* and *Adobe Illustrator — Windows Version* has the following form:

%%Creator:Adobe Illustrator(TM) *version*
%%For:*username organization*
%%Title:*document-title*
%%CreationDate:*date time*
%%DocumentProcSets:Adobe_Illustrator_*version level revision*
%%DocumentSuppliedProcSets:Adobe_Illustrator_*version level revision*
%%DocumentFonts: *font ...*
%%+*font ...*
%%BoundingBox:*llx lly urx ury*
%%TemplateBox:*llx lly urx ury*

The principal difference for the comments that convey arbitrary text information is in how the strings of characters are delimited. In Adobe Illustrator 88 documents, the individual strings are valid PostScript language strings, while in other versions the last text item in a comment is just terminated by a newline character.

The following additional comments are needed by *Adobe Illustrator — Windows Version* to provide the information that is stored in the Macintosh resource fork by the Macintosh versions of Adobe Illustrator (see Section 9, **Macintosh Resources**):

%%Template:{*filename*}
%%PageOrigin:*x y*
%%PrinterName:{*printer brand name*}
%%PrinterRect:*llx lly urx ury*

%%Template:{*filename*}

The **%%Template** comment specifies the full pathname of the template for the illustration. If no name is given, no template file is used (see Section 10.2, **TEMP Resource**). If this comment is omitted, *Adobe Illustrator — Windows Version* will put up the template dialogue box to request the name of a template file. *This is not an official structuring comment.*

%%PageOrigin:*x y*

The **%%PageOrigin** comment specifies the coordinates of the document's page origin (in points) as specified by the *Page* tool (see Section 10.3, **PAGE Resource**). If this comment is omitted**,** the page origin will be set to the ruler origin*. This is not an official structuring comment.*

%%PrinterName:{*printer brand name*}

The **%%PrinterName** comment specifies the brand name of a printer (e.g. "Apple LaserWriter.") T*his is not an official structuring comment.*

%%PrinterRect:*llx lly urx ury*

The **%%PrinterRect** comment specifies the bounding box of the image area of the printer identified in the **%%PrinterName** comment. *The coordinate system used in this comment is 4th quadrant as opposed to the 1st quadrant system used in all other structuring comments.* This means that *lly* and *ury* are measured from the top of the page instead of the bottom of the page. (Microsoft Windows® uses the 4th quadrant system and its use for this comment was a technical oversight.) This comment provides information for the page setup on the specified printer. If this comment is omitted, the default is set for letter size, portrait orientation on the Apple LaserWriter®. *This is not an official structuring comment.*

3.2 Procedure Sets

The syntax for procedure set definitions is:

```
<proc sets> ::=      {<proc set def>}*

<proc set def> ::=   %%BeginProcSet:proc_set_name level revision
                     userdict /proc_set_name size dict dup begin put
                     /initialize {
                        ...
                       } def
                       /terminate {
                        ...
                       } def
                      <other PostScript definitions>
                     currentdict readonly pop end
                     %%EndProcSet
```

Each procedure set is defined in its own dictionary using straightforward PostScript code. The dictionary is created on the first line of the definition. The dictionary must be created large enough to hold the subsequent procedure definitions. After the dictionary is created it is pushed onto the PostScript dictionary stack and becomes the first dictionary used by the PostScript interpreter for new definitions. Following the individual definitions for the procedure set, the dictionary is made read-only and then it is popped off the PostScript dictionary stack.

Each Adobe Illustrator procedure set has at least two well-known procedures defined: *initialize* and *terminate*. The *initialize* procedure is invoked by the setup section of the document script (Section 4) and the *terminate* procedure is invoked by the trailer section of the script (Section 6). **[AI88]**

4. SCRIPT SETUP

The syntax for the script setup section for an Adobe Illustrator 88 document is:

```
<setup> ::=        %%BeginSetup
                   <proc sets init>
                   <font encoding>
                   <pattern defs>
                   %%EndSetup
```

The following sections describe the individual components of the setup section of the document.

For the other versions of Adobe Illustrator, the setup section of the script pushes the main procedure dictionary onto the dictionary stack for the PostScript interpreter and may re-encode a font for use in the document. The syntax is:

```
<setup> ::=        %%BeginSetup
                   Adobe_Illustrator_version begin
                   <font encoding>
                   %%EndSetup
```

4.1 Procedure Set Initialization

For each of the procedure sets used in an Adobe Illustrator 88 document, the setup section of the document executes the *initialize* procedure that is defined by convention in each of the procedure sets used in the document. The syntax is:

```
<proc sets init> ::=    {<initialize>}*

<initialize> ::=        proc_set_name  /initialize get exec
```

The initialization instructions are straightforward PostScript code. The PostScript dictionary object that contains the definitions for a procedure set is specified first. Th*e /initialize* token is the name of a procedure stored in the dictionary. The **get** operator obtains the named procedure object from the dictionary and pushes it onto the PostScript operand stack. Finally, the **exec** operator executes the procedure that is on the top of the stack, pushed there by the preceding **get** operator.

The typical collection of procedure sets initialized in an Adobe Illustrator 88 document is:

```
Adobe_cmykcolor /initialize get exec
Adobe_cshow /initialize get exec
Adobe_customcolor /initialize get exec
Adobe_pattern /initialize get exec
Adobe_Illustrator88 /initialize get exec
```

In the script trailer part of the document, the complementary *terminate* procedure is executed for each of the procedure sets.

4.2 Font Re-encoding

In the Macintosh environment, the mapping between ASCII characters and glyphs in a font is different from the standard mapping used in a PostScript font. Therefore to print a document correctly, the mapping must be changed for each PostScript font used in an Adobe Illustrator document. The action of altering the mapping between character codes and glyphs is called *re-encoding* the font.

The syntax for re-encoding fonts in an Adobe Illustrator document is:

```
<font encoding> ::=    {<re-encoding>}*

<re-encoding> ::=      %%BeginEncoding:newfontname oldfontname
                       <Z>
                       %%EndEncoding
```

The **%%BeginEncoding** and **%%EndEncoding** comments are not official structuring comments.

The re-encoding is performed by the **Z** operator, which is defined as follows for Adobe Illustrator 88:

[*newencoding*] /*newfontname* /*oldfontname direction* **Z**

The **Z** operator creates a new font from an existing font by changing portions of the new font's encoding vector. The *newencoding* operand is an array of encoding numbers and literal glyph names organized as follows:

[$code_1$ /$name_{11}$ /$name_{12}$ ... /$name_{1j}$
$code_2$ /$name_{21}$ /$name_{22}$ ... /$name_{2k}$
...
$code_n$ /$name_{n1}$ /$name_{n2}$ ... /$name_{nl}$]

where each *code* is in the range [0,255] and each *name* is the literal glyph name. The "/" preceding each name is the syntax used to distinguish a PostScript literal name from an executable name. This array describes a set of sequences of glyph names to install in the new encoding vector. Each sequence begins with the character index of the first name to be replaced. Subsequent names are replaced up to the next character index entry in the *newencoding* array at which point a new sequence of replacement names begins, starting with the one at the new character index.

The *newfontname* and *oldfontname* operands are the PostScript names for the new font and the original font. These names must be the same as the names given in the **%%BeginEncoding** comment. The *direction* operand is zero. *For the other versions of Adobe Illustrator besides the Japanese Edition and Adobe Illustrator 88, there is no* direction *operand.*

In the following example of how the **Z** operator is used, a new font named *_Times-Roman* is derived from the original *Times-Roman* font. Three sequences of characters within the encoding vector are replaced. The first one-character sequence to be replaced is number 39, the second one-character sequence is number 96, and the third subsequence of the array replaces the characters numbered 128 and above.

```
%%BeginEncoding:_Times-Roman Times-Roman
[
39/quotesingle 96/grave  128/Adieresis/Aring/Ccedilla/Eacute/Ntilde/Odieresis
/Udieresis/aacute/agrave/acircumflex/adieresis/atilde/aring/ccedilla/eacute
/egrave/ecircumflex/edieresis/iacute/igrave/icircumflex/idieresis/ntilde
/oacute/ograve/ocircumflex/odieresis/otilde/uacute/ugrave/ucircumflex
/udieresis/dagger/.notdef/cent/sterling/section/bullet/paragraph/germandbls
/registered/copyright/trademark/acute/dieresis/.notdef/AE/Oslash
/.notdef/.notdef/.notdef/.notdef/yen/.notdef/.notdef/.notdef
/.notdef/.notdef/.notdef/ordfeminine/ordmasculine/.notdef/ae/oslash
/questiondown/exclamdown/logicalnot/.notdef/florin/.notdef/.notdef
/guillemotleft/guillemotright/ellipsis/.notdef/Agrave/Atilde/Otilde/OE/oe
/endash/emdash/quotedblleft/quotedblright/quoteleft/quoteright/.notdef
/.notdef/ydieresis/Ydieresis/fraction/currency/guilsinglleft/guilsinglright
/fi/fl/daggerdbl/periodcentered/quotesinglbase/quotedblbase/perthousand
/Acircumflex/Ecircumflex/Aacute/Edieresis/Egrave/Iacute/Icircumflex
/Idieresis/Igrave/Oacute/Ocircumflex/.notdef/Ograve/Uacute/Ucircumflex
/Ugrave/dotlessi/circumflex/tilde/macron/breve/dotaccent/ring/cedilla
/hungarumlaut/ogonek/caron
]/_Times-Roman/Times-Roman 0 Z
%%EndEncoding
```

In Adobe Illustrator Japanese Edition, the **Z** operator is as follows:

[newencoding]/newfontname/oldfontname direction fontscript **Z**

All operands are as before except the *direction* operand is 0 for horizontal writing and 1 for vertical writing. The *fontscript* operand is 0 for Roman typefaces and 1 for Kanji typefaces. Note that for composite fonts (such as Kanji fonts), the *[newencoding]* array should be an empty array.

4.3 Pattern Definition [AI88]

Patterns used by Adobe Illustrator are defined in the script setup section of the document. A pattern is essentially just another Adobe Illustrator illustration which can be drawn repeatedly. Therefore parts of the description of how patterns are defined must necessarily refer forwards in the document to the description of how an illustration is described in the document script section.

Each pattern is defined by a rectangle which is used to tile the drawing area. The illustration within the rectangle when the pattern is defined constitutes the pattern to be used when a path is filled or stroked with the pattern.

The syntax for a pattern is:

```
<pattern defs> ::=     {<pattern>}*

<pattern> ::=          %%BeginPattern: (patternname)
                       <E>
                       %%EndPattern
```

The **%%BeginPattern** *and* **%%EndPattern** *comments are not official Document Structuring Convention comments.*

The pattern definition operator, **E**, is defined as follows:

(*patternname*) *llx lly urx ury* [<layer list>] **E**

The **E** operator defines a new pattern called *patternname* using the *layer list* for which the bounding box is specified by (*llx, lly*) and (*urx, ury*).

Section 5 describes how a general illustration is composed of layers, each of which is drawn on top of lower layers which appear earlier in the sequence of layers. Because a pattern is really just a mini-illustration, it too is composed of layers. The syntax for the list of layers in a pattern is:

```
<layer list> ::=        {<layer>}*

<layer> ::=             <@>
                        <&>
```

Each layer of the pattern is defined in two parts. The first part defines the color to be used for filling and stroking the pattern. The second part defines the other style parameters and the paths for drawing the pattern.

(*colordefinition*) @

The **@** operator defines the color and overprinting style to be used for the associated layer in the pattern. The *colordefinition* parameter begins with a specification of the overprinting option (see the definition of the operators **O** and **R** in Section 5.3). The filling/stroking color is then defined using the simple gray operators (**g**/**G**), the process color operators (**k**/**K**), or the custom color operators (**x**/**X**). All of the color operators are defined in Section 5.3.

(*tiledefinition*) **&**

The **&** operator defines the *tile* for the pattern layer which includes the drawing styles and the illustration components for the layer. This is identical to the representation of objects in the document script except that the color components are specified separately and both parts of the object are specified separately as PostScript strings, which are enclosed in parentheses. The use of strings also limits the size of a pattern layer definition to 64K bytes.

If the pattern background is filled and/or stroked, the first layer of the pattern defines the background for the tile. If the pattern tile rectangle is filled, then the special "_" operator is first used to specify a fill of the tile rectangle. If the pattern tile rectangle is stroked, then normal path construction of the rectangle is used to specify the pattern tile to stroke. By breaking down the filling and stroking of the pattern tile in this manner, a performance optimization is performed when imaging. See below for an example of both filling and stroking the background pattern tile rectangle.

– _

The _ operator is used to signal to the pattern machinery that the tile rectangle for the path is to be filled with the fill color previously specified to the **@** operator.

An example pattern definition is:

```
%BeginPattern: (no vegetation)
(no vegetation) 105 561.875 138 594.875 [
(0 O 0 R 0.03 0.05 0.15 0 (PANTONE 468 CV) 0 x 0.03 0.05 0.15 0 (PANTONE 468 CV) 0 X) @
_  &
(0 O 0 R 0.125 0.25 0.525 0 (PANTONE 465 CV) 0 x 0.125 0.25 0.525 0 (PANTONE 465 CV) 0 X) @
(
0 i 0 J 0 j 1 w 4 M []0 d
%%Note:
105 561.875 m
105 594.875 L
138 594.875 L
138 561.875 L
105 561.875 L
s
) &
(0 O 0 R 0 0.15 0.4 0 (PANTONE 156 CV) 0 x 0 0.15 0.4 0 (PANTONE 156 CV) 0 X) @
(
0 i 0 J 0 j 0.3 w 4 M []0 d
%%Note:
105 599 m
105 560 l
S
108 599 m
108 560 l
S
...
138 599 m
138 560 l
S
) &
] E
%%EndPattern
```

The example above defines a pattern called “no vegetation.” This example has the pattern tile filled and stroked. First is the pattern name, the bounding box for the pattern tile, then the layer list. The first item in the layer list specifies the custom color “PANTONE 468 CV” as the fill color of the pattern tile. The “**_**” operator specifies a fill of the pattern tile. The next layer in the pattern specifies a stroke of the pattern tile. The custom color “PANTONE 465 CV” is the stoke color. Following the color specification is the drawing of the pattern tile itself. See Section 5.4 for a description of the style options selected at the beginning of the tile definition. Each path in the tile layer is then specified with a sequence of **m** (moveto) and **l** (lineto) operations. Each path in the layer is stroked by the **S** operator (Section 5.6). Once the pattern tile is drawn and stroked, the elements of the pattern follow. In this example, they are vertical lines which are all stroked with the same color.

5. SCRIPT BODY

An illustration is composed of a sequence of graphic elements. The PostScript imaging model is based on opaque ink so that elements later in the sequence are effectively “on top of” objects earlier in the sequence and so may obscure the earlier elements. The syntax for the sequence of elements is:

```
<element seq> ::=     {<element> | <import doc>}*

<element> ::=         {<A>}
                      <group> | <object>

 <group> ::=          <u>
                      <object seq>
                      <U> |
                      <q>
                      <object seq>
                      <Q>

<object seq> ::=      {<element>}*

<object> ::=          <color>
                      <graphics state>
                      {%%Note: arbitrary text}
                      <graphic> | <text>

<graphic> ::=         <path>
                      <paint operator>
```

The following sections describe the individual operators for describing graphic objects. Each section describes the operators for one of the object components shown in the syntax above.

5.1 Locked Object Operator [AI88]

flag **A**

The **A** operator specifies whether the object defined immediately after this operator can be selected when editing the document with Adobe Illustrator. The *flag* operand may be either 0 or 1. If *flag* is 1, the object is "locked" and may not be selected. If *flag* is 0, the object may be selected.

5.2 Group Operators

Several operators are provided to support Adobe Illustrator’s ability to combine a number of graphic elements into a single composite object.

– **u**

The **u** operator marks the beginning of a sequence of elements to be grouped into a composite object. All subsequent graphical elements in the script, including other groups, up to a matching **U** operator are included in the group.

– **U**

The **U** operator marks the end of a sequence of elements to be grouped into a composite object. The **U** operator must be matched by a preceding **u** operator.

– **q**

The **q** operator is similar to the **u** operator, except that the first object in the group specifies a mask (clip path) which specifies a boundary for subsequent objects in the group so that only objects within the boundary are visible when the illustration is drawn. **[AI88]**

– **Q**

The **Q** operator is similar to the **U** operator, except that it marks the end of a sequence of elements containing a mask. **[AI88]**

5.3 Color Operators

gray **g**

The **g** operator specifies the gray tint to use for filling paths. The *gray* operand must be a number between 0.0 (black) and 1.0 (white).

gray **G**

The **G** operator is similar to the **g** operator, but specifies the gray color to use for stroking paths.

cyan magenta yellow black **k**

The **k** operator is equivalent to the PostScript **setcmykcolor** operator, which specifies the process color to use for filling paths. Each operand must be a number between 0.0 (minimum intensity) and 1.0 (maximum intensity). If the **setcmykcolor** operator is not defined by the PostScript interpreter, the Adobe Illustrator prologue defines it in terms of the original **setrgbcolor** operator by transforming the operands as follows:

red = 1 - min(1, *cyan* + *black*)
green = 1 - min(1, *magenta* + *black*)
blue = 1 - min(1, *yellow* + *black*)

If the output device is monochrome, the NTSC conversion from the red-green-blue color model is used to render the color as a shade of gray. The equivalent parameter to the **g**/**G** operators is:

gray = 0.3 * *red* + 0.59 * *green* + 0.11 * *blue*.

The conversion from red, green and blue to gray is performed automatically by the PostScript interpreter for a monochrome output device.

cyan magenta yellow black **K**

The **K** operator is similar to the **k** operator, but specifies the color to use for stroking paths.

cyan magenta yellow black (*name*) *gray* **x**

The **x** operator defines a custom color to be used for filling paths. The *cyan*, *magenta*, *yellow*, and *black* operands are interpreted the same as for the **k**/**K** operators. The *gray* operand is treated the same as for the **g**/**G** operators and specifies the screen fraction of the custom color (in the range [0.0, 1.0]). The *name* operand is an valid PostScript string that is used to name the custom color. **[AI88]**

For example:

```
0.45 0 0.25 0 (PANTONE 570 CV) 0 x
```

cyan magenta yellow black (*name*) *gray* **X**

The **X** operator is similar to the **x** operator, but specifies the custom color to use for stroking paths. **[AI88]**

(*patternname*) p_x p_y s_x s_y *angle rf r k ka* [*a b c d* t_x t_y] **p**

The **p** operator specifies the pattern to be used for subsequent fill operations. The first operand names the pattern as defined in the script setup sequence (see Section 4.3). The remaining operands specify the transformations (in order) to be applied to the pattern before it is used to fill a path. The (p_x, p_y) pair specifies the offset from the ruler origin of the origin to be used for tiling the pattern. Each distance is specified in points. The s_x and s_y operands specify the scale factors to be applied to the x and y dimensions, respectively, of the pattern. The *angle* operand specifies the angle in counterclockwise degrees that the pattern is to be rotated. The *rf* operand is a flag which indicates whether to apply a reflection to the pattern. The *r* operand specifies the angle of the line from the origin about which the pattern is to be reflected if the *rf* operand is non-zero. The *k* operand specifies the shear angle. The *ka* operand specifies the angle of the shear axis if *k* is non-zero. The matrix operand specifies the initial matrix to which all of the other pattern transformations are to be applied. This matrix describes transformations that are not otherwise expressible as the single combination of the other transformations. (See the description of the *Transform Pattern Style* sub-menu of the *Paint* menu in the *Adobe Illustrator 88 User Guide*.) **[AI88]**

(*patternname*) p_x p_y s_x s_y *angle rf r k ka* [*a b c d* t_x t_y] **P**

The **P** operator is similar to the **p** operator, but specifies the stroke pattern. **[AI88]**

flag **O**

The **O** operator specifies whether overprinting will be used when a path is filled. If *flag* is 1, overprinting will be used, otherwise *flag* must be zero. See *Adobe Illustrator 88 Color Guide* (part of the Adobe Illustrator 88 package) for a discussion of overprinting in Adobe Illustrator 88 documents. **[AI88]**

flag **R**

The **R** operator is similar to the **O** operator, but specifies whether overprinting will be used when a path is stroked. **[AI88]**

5.4 Graphics State Operators

The components of the graphics state for a PostScript language interpreter are defined in section 4.3 of the *PostScript Language Reference Manual*. For Adobe Illustrator some of the graphics state components may be set by the following operators:

[*array*] *phase* **d**

The **d** operator is equivalent to the PostScript **setdash** operator, which sets the dash pattern parameter in the graphics state, controlling the dash pattern used when paths are subsequently stroked. The array of values specifies distances in user space along the path for the length of dashes and gaps, respectively, in the pattern. The phase operand determines the phase of the dash pattern with respect to the start of the path. It is specified as a distance in user space into the pattern at which to begin the marking of the path. Note that Adobe Illustrator does not provide an interface for users to adjust this phase parameter. Adobe Illustrator will preserve this phase for documents which the user edits and then saves. The initial dash pattern is a solid line.

flatness **i**

The **i** operator is equivalent to the PostScript **setflat** operator, which sets the flatness parameter in the graphics state. The *flatness* parameter specifies the accuracy or smoothness with which curves are rendered as a sequence of straight line segments. Specifically, it sets the maximum distance in device pixels that is permitted between the mathematical path and a given straight line segment. The default value for the flatness parameter is 1.0. If *flatness* is specified as 0, the flatness is set to the flatness parameter in effect when the Adobe Illustrator prologue was executed.

linejoin **j**

The **j** operator is equivalent to the PostScript **setlinejoin** operator, which sets the current line join parameter in the graphics state to *linejoin*. This parameter specifies the shape to put at corners in paths when they are stroked. The *linejoin* parameter may be 0 for mitered joins, 1 for round joins, and 2 for beveled joins. The initial line join is 0 for mitered joins.

linecap **J**

The **J** operator is equivalent to the PostScript **setlinecap** operator, which sets the current line cap parameter in the graphics state to *linecap*. If *linecap* is 0, butt end caps are used and the end of a line is squared off. If *linecap* is 1, round end caps are used. If *linecap* is 2, projecting square end caps are used. The projection extends beyond the end of the line by a distance which is half the line width. The initial line cap is 0 for square butt ends.

miterlimit **M**

The **M** operator is equivalent to the PostScript **setmiterlimit** operator, which sets the miter limit parameter in the graphics state to *miterlimit*. The operand must be a number greater than 1. When mitered line joins have been specified and two line segments meet at a sharp angle, it is possible for the miter to extend far beyond the thickness of the line stroking the path. The miter limit imposes a limit on the ratio of the length of the miter to the line width. When the limit is exceeded, a bevel join is used instead of a miter. The initial miter limit is 10.

linewidth **w**

The **w** operator is equivalent to the PostScript **setlinewidth** operator, which sets the line width parameter in the graphics state to *linewidth*. This parameter controls the thickness of the line used to stroke a path and is specified as a distance in user space. The initial line width is 1.0.

5.5 Path Construction Operators

Drawing in the PostScript language is accomplished by constructing a path and then filling or stroking it. This section defines the path construction operators. Only one path (the *current path*) may be constructed at a time. The current path is initially empty and is reset to empty by the painting operators. A path is constructed by appending segments which are either straight lines or Bézier curves. The last point on a segment is termed the *current point* and new segments are appended to the current point. The first operator on a path must be **m** to establish an initial *current point*.

As each new segment is appended to the path, the new current point is marked as either a *smooth point* or a *corner point*. If the point is marked smooth, then the point and the two associated Bézier direction points of the segments connected by the point are assumed to be collinear. If the point is marked as a corner point, no constraint is assumed. A straight line segment may be thought of as a degenerate Bézier curve in which the direction points are coincident with the end points.

The syntax for a path, which requires an **m** operator as its first component, is:

```
< path> ::=          <m>
                     {<path operator>}*
```

The path operators are defined as follows:

x y **m**

The **m** operator is equivalent to the PostScript **moveto** operator. It changes the current point to (x, y) omitting the connecting line segment.

x y **l**

The **l** operator appends a straight line segment from the current point to (x, y). The new current point is a smooth point.

x y **L**

The **L** operator is similar to the **l** operator, but the new current point is a corner.

$x_1\ y_1\ x_2\ y_2\ x_3\ y_3$ **c**

The **c** operator appends a Bézier curve to the path from the current point to (x_3,y_3) using (x_1,y_1) and (x_2,y_2) as the Bézier direction points. The new current point is a smooth point.

$x_1\ y_1\ x_2\ y_2\ x_3\ y_3$ **C**

The **C** operator is similar to the **c** operator, but the new current point is a corner.

$x_2\ y_2\ x_3\ y_3$ **v**

The **v** operator adds a Bézier curve segment to the current path between the current point and the point (x_3, y_3), using the current point and then (x_2, y_2) as the Bézier direction points. The new current point is a smooth point.

$x_2\ y_2\ x_3\ y_3$ **V**

The **V** operator is similar to the **v** operator, except that the new current point is a corner.

$x_1\ y_1\ x_3\ y_3$ **y**

The **y** operator adds a Bézier curve segment to the current path between the current point and the point (x_3, y_3), using (x_1, y_1) and then (x_3, y_3) as the Bézier direction points. The new current point is (x_3, y_3) and it is a smooth point.

$x_1\ y_1\ x_3\ y_3$ **Y**

The **Y** operator is similar to the **y** operator, except that the new current point is a corner.

5.6 Painting Operators

Each of the path painting operators consumes the *current path* and resets it to empty unless otherwise stated in the operator descriptions below.

– N

The **N** operator neither fills nor strokes the current path (see the **F**/**f** and **S**/**s** operators). Paths that are invisible in the final document may be used for templates, alignment marks, and so on while using Adobe Illustrator to edit a document.

– n

The **n** operator is similar to the **N** operator, but first closes the current path by appending a straight line segment to the current path from the current point to the beginning point of the path.

– F

The **F** operator fills the area enclosed by the current path with the current filling color or pattern. The inside of the current path is determined by the PostScript non-zero winding rule. (See Section 4.6 of the *PostScript Language Reference Manual*.)

– f

The **f** operator is similar to the **F** operator, but first closes the current path by appending a straight line segment to the current path from the current point to the beginning point of the path.

– S

The **S** operator strokes the current path with a line using the current stroking color or pattern. The line width is specified by the graphics state (see the **w** operator) and the line is centered on the path with its sides parallel to the path. The joins between path segments are specified by the line join parameter in the graphics state (see the **j** operator) and the ends of the path segments or dash lines within a segment (see the **d** operator) are specified by the end cap parameter of the graphics state (see the **J** operator).

– s

The **s** operator is similar to the **S** operator, but first closes the current path by appending a straight line segment to the current path from the current point to the beginning point of the path.

– B

The **B** operator is similar to the **F** operator, but does not reset the current path to empty.

– b

The **b** operator is similar to the **f** operator, but does not reset the current path to empty.

– H

The **H** operator neither fills nor strokes the current path, but does not consume the path. This is effectively a no-op. **[AI88]**

– h

The **h** operator is similar to the **H** operator, but first closes the current path by appending a straight line segment to the current path from the current point to the beginning point of the path. **[AI88]**

– W

The **W** operator intersects the current clip path in the graphics state with the current path and sets a new, reduced clip path in the graphics state. No marks are made outside the area enclosed by the current clip path by subsequent fill and stroke operations. This operator is used to establish a mask. **[AI88]**

5.7 Text Operators

Adobe Illustrator text blocks are rendered in three steps. First the font is selected. Second a new user space is defined by concatenating a matrix with the current transformation matrix and then the individual lines of a text block are imaged. The syntax for a text block is:

```
<text> ::=          <z>
                    <a> | <e> | <I> | <o> | <r>
                    {<t>}*
                    <T>
```

The following operators are used to render text blocks:

/fontname size leading kerning align **z**

The **z** operator selects a new *current font* to be used in rendering subsequent text blocks. The *fontname* operand is the name of the re-encoded PostScript font, such as _Times-Roman (see Section 4.2). The *size* operand specifies the scaled size of the font in user space units. Similarly, the non-negative *leading* operand specifies the vertical spacing between successive lines of text (in user space units). The *kerning* operand specifies a distance in user space to be added to the X width of each character, thereby modifying the spacing between characters. This value is stored in the Adobe Illustrator graphics state for use by the **t** operator, which is described below. The *align* parameter specifies how successive lines of text are aligned with each other. An *align* value of 0 specifies align left (flush left, ragged right text), a value of 1 specifies align center (ragged left and ragged right), and a value of 2 specifies align right (flush right, ragged left text). For Adobe Illustrator Japanese Edition, the *align* operand has additional values. An *align* value of 3 specifies vertical text flush at top and ragged at bottom, an *align* value of 4 specifies vertically centered text, and an *align* value of 5 specifies vertical text flush at bottom and ragged at top.

[$a\ b\ c\ d\ t_x\ t_y$] **a**

The **a** operator begins a text block in which individual characters are stroked with the current stroking color after they have been filled with the current filling color. The matrix operand is concatenated with the current transformation matrix to establish a new user space (see Section 4.4 of the *PostScript Language Reference Manual*). This transformation handles the arbitrary rotation and reflection of the text. The new user space establishes the origin for the first line of text.

[$a\ b\ c\ d\ t_x\ t_y$] **e**

The **e** operator is similar to the **a** operator, except that individual characters are filled with the current filling color.

[*a b c d* t_x t_y] **I**

The **I** operator is similar to the **a** operator, except that the outlines of the individual characters are appended to the *current path* instead of being rendered directly (see Section 5.5). **[AI88]**

[*a b c d* t_x t_y] **o**

The **o** operator is similar to the **e** operator, except that individual characters are neither filled nor stroked.

[*a b c d* t_x t_y] **r**

The **r** operator is similar to the **a** operator except that the individual characters are stroked with the current stroking color. The stroke line is centered on the character outline with sides parallel to the outline path. The stroke is drawn using the current graphics state parameters for the line width, line join, line cap, and dash pattern.

length (*string*) **t**

The **t** operator renders the *string* of characters on the output device. The starting point for the string, (S_x, S_y), is defined for each of the alignment methods as follows:

Align left	(0, 0)
Align center	($-w_x/2$, $-w_y/2$)
Align right	($-w_x$, $-w_y$)

where w_x and w_y are the sum of the individual character X widths and Y widths, respectively. The *kerning* parameter is added to each X width before it is included in the sum.

After the characters in *string* have been rendered, the user space origin is translated by the current *leading* in the negative Y direction to establish the origin for the next line of text.

Note:
The length *operand is required only in Adobe Illustrator 88 documents. Other versions of Adobe Illustrator documents must not have a length operand.*

– **T**

The **T** operator ends a block of text and restores the user space to its state prior to executing the preceding text block beginning operator (**a**, **e**, **I**, **o**, **r**).

The following example illustrates the use of the text operators:

```
/_Helvetica 12 10 0 2 z
[1 0 0 1 183 254]e
11 (Right align)t
T
```

In this example, a re-encoded version of the Helvetica font (_Helvetica) is selected at a size of 12 points with 10 point leading and no additional X kerning. The text is right aligned as specified by the **e** operator. The string is 11 characters long.

5.8 Imported Document Operators [AI88]

The syntax for importing a document into an Adobe Illustrator 88 document is:

```
<import doc> ::=      <‘>
                      %%IncludeFile:filename
                      <~>
```

The *filename* specified in the *%%IncludeFile* comment must be the same as that specified for the **‘** operator (see below). The file importing operators are:

[*a b c d* t_x t_y] *llx lly urx ury* (*filename*) **‘**

The **‘** operator specifies that the document stored in *filename* is to be imported into the illustration. The *filename* string is the full pathname for the file in the operating system’s filesystem. The matrix operand is concatenated with the current transformation matrix to establish a new user space and an origin for the imported document (see Section 4.4 of the *PostScript Language Reference Manual*). The matrix handles any rotation and reflection to be applied to the imported document. The bounding box from the imported document is specified by the *llx, lly, urx, ury* operands. The bounding box is specified by the **%%BoundingBox** comment in the imported document’s prologue. In addition to establishing a new user space, the **‘** operator redefines the PostScript **showpage** operator as a no-op and disables overprinting.

– ~

The ~ operator restores the user space in effect when the preceding **‘** operator was executed and restores the previous definition of **showpage**. (The **‘** operator executes the PostScript **save** operator to preserve the interpreter state before importing the document and the ~ operator executes the **restore** operator to return the interpreter state to that prior to executing the **‘** operator.)

Adobe Illustrator 88 modifies its **%%DocumentFonts** comment to include any fonts in the **%%DocumentFonts** comment of documents placed (included) into Adobe Illustrator.

Included files can also be saved directly into an Adobe Illustrator 88 document. If that is the case, then the **%%IncludeFile** comment described above is replaced by:

```
%%BeginDocument:filename
....  included file here
%%EndDocument
```

6. SCRIPT TRAILER

The syntax for the script trailer of an Adobe Illustrator 88 document is:

```
<trailer> ::=          %%Trailer
                       {<terminate>}*

<terminate> ::=        proc_set_name /terminate get exec
```

The procedure set termination is expressed in pure Postscript code. The PostScript dictionary object that contains the procedure set is pushed onto the PostScript operand stack. The */terminate* token is the name of a procedure stored in the dictionary. The **get** operator obtains the named procedure from the dictionary and pushes it onto the operand stack. Finally, the **exec** operator executes the procedure that is on the top of the stack, pushed there by the preceding **get** operator.

For each procedure set that was initialized in the script setup, the trailer invokes its *terminate* procedure:

```
Adobe_Illustrator88 /terminate get exec
Adobe_pattern /terminate get exec
Adobe_customcolor /terminate get exec
Adobe_cshow /terminate get exec
Adobe_cmykcolor /terminate get exec
```

Note that the procedure sets are terminated in the opposite order from that used to initialize them. For the other versions of Adobe Illustrator, the <terminate> is:

```
<terminate> ::=        _E end
```

This trailer executes the procedure named **_E** from the *Adobe_Illustrator_version* procedure set dictionary. The **_E** procedure is responsible for printing messages that describe any error that may occur during the execution of the illustration document. The **end** operator removes the *Adobe_Illustrator_version* dictionary from the stack of dictionaries in use by the PostScript interpreter. This **end** matches the **begin** operator used in the setup section of the document.

7. DOCUMENT SYNTAX SUMMARY

```
<document> ::=<prologue>
<script>

<prologue> ::=%!PS-Adobe-2.0 EPSF-1.2
<prologue body>
%%EndProlog

<prologue body> ::= <header>
<procedure sets>

    <header> ::=            <header comments>
                            %%EndComments

    <proc sets> ::=         {<proc set def>}*

    <proc set def> ::=      %%BeginProcSet:name level revision
                            userdict /name size dict dup begin put
                            /initialize {
                               ...
                              } def
                              /terminate {
                               ...
                              } def
                             <other PostScript definitions>
                            currentdict readonly pop end
                            %%EndProcSet

    <script> ::=            <setup>
                            <script body>
                            <trailer>

    <setup> ::=             %%BeginSetup
                            <proc sets init>
                            <font re-encoding>
                            <pattern defs>
                            %%EndSetup

    <proc sets init> ::=     {<initialize>}*

    <initialize> ::=        proc_set_name  /initialize get exec

    <font encoding> ::=     {<re-encoding>}*

    <re-encoding> ::=       %%BeginEncoding:newfontname oldfontname
                            <Z>
                            %%EndEncoding

    <script body> ::=       {<element> | <import doc>}*

    <element> ::=           {<A>}
                            <group> | <object>

     <group> ::=            <u>
                            <object seq>
                            <U> |
                        <q>
                        <object seq>
                        <Q>

<object seq> ::=        {<element>}*

<object> ::=            <color>
                        <graphics state>
                        {%%Note: arbitrary text}
                        <graphic> | <text>

<graphic> ::=           <path>
                        <paint operator>

<text> ::=              <z>
                        <a> | <e> | <I> | <o> | <r>
                        {<t>}*
                        <T>

<import doc> ::=        <‘>
                        %%IncludeFile:filename
                        <~>

<trailer> ::=           %%Trailer
                        {<terminate>}*

<terminate> ::=         proc_set_name /terminate get exec
```

8. COMPRESSING ILLUSTRATOR DOCUMENTS

The standard Adobe Illustrator document prologue can consume between 6K and 11K bytes of data in the illustration document file. This can account for significant overhead both in transmitting files between users and in storing large collections of illustrations. Therefore it is often desirable to minimize the size of an Adobe Illustrator document. This section describes explicitly how to remove parts of an Adobe Illustrator document that can be restored automatically at a later time. The information in this section is presented implicitly in the body of this document, but is collected here explicitly for convenience.

Adobe Illustrator documents are usually completely self-contained. If an illustration imports another document, Adobe Illustrator has an option for copying the entire imported document's PostScript page description into the illustration document file instead of maintaining just a filename reference to the other document (see Section 5.8). If the imported document is another Adobe Illustrator document, the procedures for removing parts of a document may be applied to the imported document as well.

8.1 Removing Procedure Sets

The procedure sets used in a document are identified in the prologue header (Section 3.1) with **%%DocumentProcSets** comments. A procedure set may be removed without destroying the integrity of the document with respect to the *Document Structuring Conventions*, provided that it is replaced with the appropriate **%%IncludeProcSet** comment. To remove a procedure set, the associated **%%DocumentSuppliedProcSets** comment should be changed to a **%%DocumentNeededProcSets** comment. Then the comment **%%BeginProcSet**, the actual procedure itself, and the **%%EndProcSet** comment are replaced by the comment **%%IncludeProcSet** *name version revision*, where *name*, *version*, and *revision* are that of the procedure set being removed. Once this is done, the remaining **%%DocumentNeededProcSets**, and **%%IncludeProcSet** comments can be used to identify what was removed and where to insert it at a later time.

***Note**: If these changes are made to documents, they will not print unless there is intelligent post-processing software which reads the Document Structuring Conventions comments and provides the missing procedure sets. Adobe Illustrator itself can open these files and rewrite them appropriately for printing.*

Procedure sets may be downloaded to a PostScript printer just like fonts. Therefore, the procedure sets may be transmitted to a printer just once and then initialized, used, and terminated repeatedly to avoid the overhead of transmitting them repeatedly for a document that contains many illustrations.

A small savings may be made by removing the one line entry in both the script setup and trailer sequences that initialize and terminate the procedure sets. Beware, however, that the order of the procedure sets in the prologue header **%%DocumentProcSets** comments is not guaranteed to be the same as the order in which the procedure sets are initialized and terminated. The **%%DocumentProcSets** comments may be re-ordered, of course, to carry this information. Removing the lines from the setup and trailer sequences destroys the integrity of the document with respect to the *Document Structuring Conventions*. This means that typical spooling mechanisms that can insert procedure sets based only on the **%%DocumentProcSets** comments found in a PostScript page description file will not restore the setup and trailer information automatically. If these changes are made, only software familiar with Adobe Illustrator files (including Adobe Illustrator) will be able to reconstruct the original files.

8.2 Removing Font Re-Encodings

The re-encoding of PostScript fonts which use the *standard* Macintosh encoding may be removed and will be restored automatically by Adobe Illustrator when the file is read in and saved again. Note that documents without this explicit re-encoding of the font cannot be printed directly. The code that re-encodes a font appears between **%%BeginEncoding** and **%%EndEncoding** comments in the script setup part of the document. The array defining the new encoding should be compared with the standard one if there is a possibility of a different encoding being supplied by an application other than Adobe Illustrator. If the new encoding is standard, the entire **Z** operator and its operands may be removed. The trailing **%%EndEncoding** comment may be removed as well since it carries no additional information.

8.3 Removing Other Comments

Section 3.1 which describes the header comments, indicates which comments are always required and those which are necessary if certain Adobe Illustrator features are used. These comments are essential if the document is to be manipulated again by Adobe Illustrator. All of the other comments are ignored by Adobe Illustrator and will be inserted into the document the next time it is saved to disk. The comments required in all documents are **%%BoundingBox** and **%%TemplateBox**. The other comments read by Adobe Illustrator are: **%%DocumentCustomColors** and **%%CMYKCustomColor**. It would be wise to check any software that will manipulate reduced Adobe Illustrator files to determine which of the remaining header comments should be retained.

The **%%Note** comments that follow the graphics state operators in an Adobe Illustrator document are preserved by Adobe Illustrator when it edits a document. However, the comments may be eliminated if the document is frozen and will not be changed again.

9. CREATING ILLUSTRATOR DOCUMENTS

Other applications may create documents to be edited subsequently by Adobe Illustrator. Very few of the prologue, setup and trailer components are needed by Adobe Illustrator to be able to read a document. Note that these documents cannot be printed until they are opened and re-saved within Adobe Illustrator. The following example demonstrates a minimal document that is accepted by Adobe Illustrator 88:

```
%!PS-Adobe-2.0 EPSF-1.2
%%BoundingBox:72 72 154 154
%%TemplateBox:0 0 612 792
%%EndComments
%%EndProlog
0 G
72 72 m
154 72 L
154 154 L
72 154 L
72 72 L
s
/_Times-Roman 12 10 0 1 z
[1 0 0 1 108 108]e
6 (Line 1)t
6 (Line 2)t
T
%%Trailer
```

This example illustration draws a one inch square that is one inch in from the bottom left-hand corner of page “5” of the default Adobe Illustrator drawing area. First the stroking color is set to black (“0 G”). The box is constructed with one **moveto** (**m**) and three **lineto** (**L**) operations and then it is stroked with the **s** illustration operator. In the middle of the square, two lines of centered text (“Line 1” and “Line 2”) are drawn in the Times-Roman font, which is referred to by its re-encoded name, “_Times-Roman” (see Section 5.7).

The example file is only 235 bytes long, but with all of the Adobe Illustrator prologue and script overhead, the file is approximately 12K bytes long. Almost all of the additional information may be inserted automatically to construct a file that may be printed by a PostScript printer (see Section 8).

Section 10.2 below describes the format for the document template information required by Adobe Illustrator on the Macintosh. When opening a document without this template information, Adobe Illustrator will bring up a dialog prompting the user for the location of the template for this file (if any). By supplying this resource information directly in your minimal documents, you may avoid this inconvenience.

10. MACINTOSH RESOURCES

On the Macintosh, the resource fork of an Adobe Illustrator document contains several ancillary resources. These resources are described in the following sections.

10.1 PICT Resource

An Adobe Illustrator document may have a graphical screen representation provided so that a preview of the illustration may be manipulated on the screen by a page composition system. On the Macintosh this representation is saved as a QuickDraw PICT picture resource within the resource fork of the document. The resource is given a resource type of *PICT* and a resource number of 256.

The picture's *picFrame* bounding box matches the bounding box of the illustration. This bounding box is specified by the **%%BoundingBox** comment in the prologue header. That is the width and height of the *picFrame* are the same as the width and height, respectively, of the bounding box.

The picture resource is composed of two bitmap images: the image itself and its mask. If a particular bit is set in the mask, then the illustration has actually painted the corresponding bit in the image. Otherwise the corresponding bit has not been painted and hence should be transparent.

The mask is placed in the picture first in the QuickDraw *srcBic* mode. It punches a white hole in just those areas that are painted. Then the image is placed in the QuickDraw *srcOr* mode, which fills in the punched areas, but leaves the other areas untouched.

10.2 TEMP Resource

This resource identifies the name of the document's template file, if it has one. The resource has three components which are (in order):

- A 32-bit integer containing the directory identifier of the folder containing the template file. The integer is zero if the document has no template.
- A Pascal string containing the name of the volume on which the template file resides. The string is empty if the document has no template.
- A Pascal string containing the name of the template file itself. The string is empty if the document has no template.

The resource is given a type of *TEMP* and a resource number of 256.

10.3 PAGE Resource

This resource contains the *x* and *y* coordinates of the document's page origin as specified by the *Page* tool. The coordinates are in the default user coordinate system. In this system the unit length along both axes is 1/72". The resource consists of two 32-bit fixed point numbers. The first specifies the *y* coordinate and the second specifies the *x* coordinate. The resource is given a type of *PAGE* and a resource number of 256.

10.4 PREC Resource

This resource contains the standard 120 byte Macintosh Printing Manager print record. It describes the document's user specified printing preferences selected from the Page Setup and Print dialogs. The resource is given a type of *PREC* and a resource number of 256.

APPENDIX A: CHANGES SINCE EARLIER VERSIONS

Changes since December 20, 1989 version

- The descriptions for the operators **o** and **a** in section 5.7 have been corrected. (They were reversed.)