POSTSCRIPT™
Software From Adobe

Supporting Downloadable PostScript Language Fonts

Adobe Developer Support

Technical Note #5040

31 March 1992

Adobe Systems Incorporated

Corporate Headquarters
1585 Charleston Road PO Box 7900
Mountain View, CA 94039-7900
(415) 961-4400 Main Number
(415) 961-4111 Developer Support
Fax: (415) 961-3769

Adobe Systems Eastern Region
24 New England
Executive Park
Burlington, MA 01803
(617) 273-2120
Fax: (617) 273-2336

Adobe Systems Europe B.V.
Europlaza
Hoogoorddreef 54a
1101 BE Amsterdam Z-O, Netherlands
+31-20-6511 200
Fax: +31-20-6511 300

Adobe Systems Japan
Swiss Bank House 7F
4-1-8 Toranomon, Minato-ku
Tokyo 105, Japan
81-3-3437-8950
Fax: 81-3-3437-8968

PN LPS5040

Contents

Supporting Downloadable PostScript Language Fonts

1 Introduction

This document is an overview of the issues involved in using downloadable font software with PostScript™ interpreters. The discussion is not environment-specific; that is, the principles outlined should apply in any environment that supports downloadable PostScript language font programs. Specific details are provided for the Macintosh® and MS-DOS® environments for which Adobe currently produces PostScript language font programs.

Adobe Systems produces and distributes PostScript language font programs, but the mechanisms to support them are provided by the manufacturer of the operating system. Questions about specific environments should be directed to the creators of those environments. In particular, if you want to know more about the Macintosh printing environment, check with Apple; if you want know the details of Microsoft® Windows™, ask Microsoft.

This document (and the Adobe Systems Developers Association) will help you determine what to do at the PostScript language level to support downloadable font software, and generally what should be done on the supporting host to facilitate this.

In this document, unless otherwise specified, font is used to refer to a downloadable font program. These fonts differ from the screen fonts displayed on the user's screen.

2 Downloadable Font Programs

A downloadable font is an executable PostScript language program. In particular, it is an ASCII text file transmitted to the PostScript interpreter; the file is executed by the interpreter as it is received.

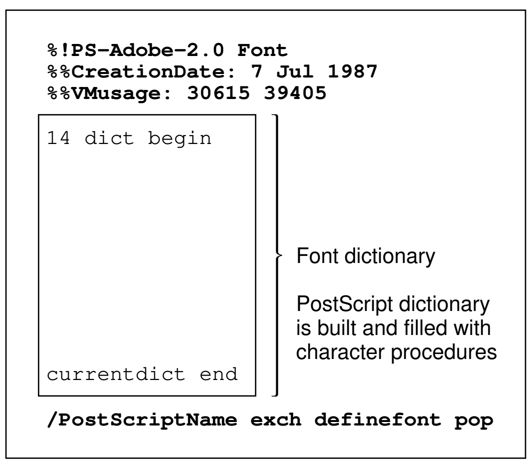
There are two tasks performed during the execution of a font program:

- Dictionary construction: a PostScript language dictionary is constructed, which includes the set of procedures for drawing each character in the font, plus encoding and metric information.

- Invocation of **definefont**: This is typically the last line in a downloadable font program. It contains an instance of the **definefont** operator, which takes a font name and the dictionary built by the code described above, and registers the font in a dictionary known as **FontDirectory**.

The memory allocated for these dictionaries comes from the same “pool” of memory as all PostScript language objects—there is no special “font memory.” It is not until the final invocation of **definefont** that the dictionary is assigned the semantics of a font dictionary.

Figure 1 *Structure of a font program*



3 Font File Formats

A PostScript language font program should be a 7-bit ASCII data stream when it is sent to a PostScript interpreter. However, the programs are not always stored this way on the host system. In environments where disk space is a concern, the files are compressed by some scheme to reduce their size on the host system, but they need to be de-compressed before they can be understood by a PostScript interpreter.

3.1 Hexadecimal Format

PostScript language font programs from Adobe Systems are encrypted. That is, most of the actual program file has been reduced to an unreadable form that is de-coded by the PostScript printer before it is executed. The encrypted data is in hexadecimal form, as a stream of digits. These digits are preceded by clear-text PostScript language code, and might have a line or two of clear-text PostScript language at the end as well. The following is an example of part of a font program:

```
currentfile eexec
8efbfe33574cfc5de0271b075aaa68c00c5eeb7ad2c35f523b627e47e8373cab1a18
205bd59f9f2ed6fd769bfe720c3a8f69ca767120359e05ac7fad673d45062159faa5
411dd62d8b1b6e7f51bba60dd6c2a9aaabbfe7186
   % ... more hex
fa6ac9a06028d13e3e8e62554992324bb30fe553bad2149765664cdeb3dd025b96f3
5ecd9d62a04c7939eb226b0dd2149765664cdeb318d56110e
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000000000000
cleartomark
```

This hexadecimal representation accounts for a large portion of the size of a font file. Because there is no advantage to having this in human-readable form, it is often packed into a binary representation on the host system. (The relationship between hex and binary is two-to-one; it takes two bytes of hex to represent eight bits of data.)

There are currently two binary formats for fonts released by Adobe Systems. The first conforms to the specification by Apple Computer to store the fonts in POST resources. The second format is used only in the IBM® PC® environment, and the compression scheme is one implemented by Adobe Systems.

To download a font file, it must be converted first from its compressed form and transmitted as pure 7-bit ASCII hexadecimal to the PostScript interpreter. Typically, this is done as the font file is being read from the disk.

3.2 Macintosh ‘POST’ Resources

There are six kinds of POST resources used to store PostScript language font programs. Each of them has a type identifier (from 0 to 5) as the first byte in the resource. This byte defines which of the POST resources it is. The rest of the resource is determined as follows:

- TYPE 0: This resource is a comment. It should not be downloaded to a PostScript interpreter.
- TYPE 1: ASCII text. This is the resource type used for the human readable clear-text portions of the font program. These can be sent to the PostScript interpreter directly.
- TYPE 2: Hexadecimal data. This data is actually stored as binary, and must be converted into hex as it is being transmitted to the PostScript interpreter.
- TYPE 3: End-of-file marker. When a TYPE 3 resource is encountered, an end-of-file (EOF) indication must be sent to the PostScript interpreter. This depends on the communications channel, but is typically a control-D on a serial connection and a special EOF packet on AppleTalk®.
- TYPE 4: Font program is in data fork. This resource type indicates that the remaining portion of the font program is stored in the data fork of the Macintosh file, and should be read directly from there. No further resources should be read, and an end-of-file should not be transmitted at the end of the data segment unless it is the end of the print job in which the font file is downloaded. Adobe does not currently use this type.
- TYPE 5: End-of-font program. This resource type marks the end to the font program as stored as resources. When a TYPE 5 resource is encountered, downloading the font program should be terminated. Note that an end-of-file indication should not be sent on this resource type. Only TYPE 3 resources represent an end-of-file marker.

PostScript language font programs stored in these Macintosh resources always begin with resource ID 501, and continue sequentially until the program is exhausted (or until a TYPE 5 resource is encountered). A downloading program should consider the first TYPE 5 resource or the first sequential resource ID that cannot be located as the end of the font. (An example of the latter would be if resource ID 517 cannot be found, then 516 was the end of the font.) More information on this format can be obtained by contacting Apple Computer.

3.3 IBM PC Format

PC fonts are stored in a compressed binary format similar to the Macintosh resource format. These files can be unpacked as they are being downloaded to the PostScript interpreter. The file is conceptually divided into segments, each of which has a small header containing a “type” field and length information. There are three types of segments:

- TYPE 1: ASCII text. This text can be sent directly to the printer without any decompression.

- TYPE 2: This is binary data that should be converted to hexadecimal and transmitted to the printer as a stream of ASCII hex data (new lines can be inserted anywhere in hex streams, if necessary).

- TYPE 3: End-of-file indication. This is a flag that indicates that the end of the data segment has been reached.

Table 1 shows the header for a binary segment.

Table 1 *Header for a binary segment*

Byte Position	*Description*
Byte 1:	always 128 (decimal)
Byte 2:	integer segment type (see above)
Bytes 3–6:	length of data segment, with byte 6 as the most significant byte.
Bytes 7–length:	data segment

There can be any number of segments in the file; lengths can be odd numbers, and there can be several sequential segments of the same type.

The following are examples of data segments (all numbers are decimal):

```
128 1 20 0 0 0 <20 bytes of ASCII>

128 2  10 1 0 0 <266 bytes of binary data>

128 2  30 2 0 0 <542 bytes of binary data>

128 3 <end-of-file>
```

4 Downloading Font Programs

A PostScript language font program can be downloaded at any time during the execution of the user program. Because it is a self-contained PostScript language program, it can be inserted within a user program without disturbing the state of the interpreter. Also, because a font program allocates memory like any other program, the implicit **save** and **restore** that are performed between print jobs apply to the memory used by fonts as well as the other memory used by a PostScript language program. This means that a downloaded font normally will not persist from one print job to another (although the individual entries in the font cache will not be disturbed).

The process of actually downloading the fonts is the same whether they are downloaded in-line or before the entire document. The disk file that contains the font must be located, and the contents of the file transmitted to the printer like any other PostScript language program is transmitted (modulo any necessary decompression of the contents of the files).

4.1 Downloading During a Single Print Job

Fonts can be downloaded on an as-needed basis for each print job. In this case, it is the job of the printing manager to determine which fonts are required by a particular print job, and to download them appropriately. The fonts will not be stored from one job to another, and must be downloaded each time they are needed. In this sense they can be thought of as PostScript language definitions that must be made available to the print job for it to execute correctly. See section 4.3, “Finding the Right Fonts,” for more information on determining which fonts must be downloaded.

The act of printing a document should always be de-coupled from the original act of creating it (or composing it). A composition program should produce a PostScript language print file that can use several fonts, but might not know in advance which of the fonts will be available when the document is actually printed.

The job of managing fonts is passed to the *printing manager*, which might be part of the same application and which in many instances might also be the *print spooler*. Once a printer has been chosen, printing the document requires a few printer-specific decisions, including selecting the appropriate paper tray and perhaps providing the necessary fonts.

4.2 “In-Line” Fonts Versus “Pre-Loaded” Fonts

As long as a font definition is made available prior to its being used, a document can print successfully. The font definition can be done in-line, and just prior to use, or, in many instances, it is useful to pre-load any fonts needed for a print job. In this case, the printing manager must determine what the total list of fonts used by a document might be, and then determine which of the fonts must be downloaded. These can then be downloaded one after another just prior to sending the document file to be printed.

Note *Both the fonts and the document file must be sent together as part of the same print job. This means that no end-of-file indications are transmitted between them, and that they follow one another closely enough that no* ***timeout*** *errors are provoked.*

4.3 Finding the Right Fonts

There are three steps to printing a PostScript language document.

1. Document composition, where the user can choose to format a document in any number of different fonts (perhaps based on screen fonts, if the system uses WYSIWYG technology).

2. Translation into the PostScript language. Either the user application (for instance, a word processor) or a system-software translator (like the Macintosh Print Manager, which translates QuickDraw™ into PostScript language code) must write a PostScript language program that will correctly render the user’s document. At that stage, the correct names for the fonts must be determined. If the fonts were not originally specified by their PostScript language names (perhaps the screen fonts follow a different naming convention) then the correct names must be determined, either algorithmically derived from the user-specified names or, more likely, looked up in a name mapping table.

3. Printing the document. At this stage, the names of the fonts needed should already have been determined. The printing manager must determine which of the named fonts are already resident in the printer, and which need to be downloaded for the current print job.

The list of fonts used by a document can usually be determined by examining the comments at the beginning of the document file, if it is a conforming PostScript language document. The two document structuring conventions (DSC) comments %%DocumentNeededResources:fonts and %%DocumentSuppliedResources:fonts provide a list of all fonts used by the document. (The notation (atend) is permitted, which means that the list is to be found at the end of the file instead of the beginning.) More information on these comments can be obtained from Appendix G, “Document Structuring Conventions,” in the *PostScript Language Reference Manual, Second Edition.*

The list of fonts already resident in the printer can be obtained by querying the printer. This involves sending down a short PostScript language program that will return a list of font names across the communications channel. Because PostScript interpreters might have disks, cartridges, and even font servers, the appropriate PostScript language segment for the font query should be obtained from the PostScript printer description (PPD) file for the chosen printer. See the *PostScript Printer Description File Format Specification* for further information on these files.

5 Font Memory Management

If many fonts are used, they might need to be managed using **save** and **restore**. This might require a printing manager to download fonts in-line with the rest of the document file, as it is being downloaded. This is noted in the document file through the use of two comments.

The %%DocumentNeededResources:font comment, is found in the beginning of the file to indicate that there might be a need to download fonts in-line somewhere. Then, at the point in the document file where the font should be downloaded, there will be an instance of the %%IncludeResource:font comment, with the name of the font provided. Again, see the *Document Structuring Conventions* for more information on these comments.

It is up to the application generating the PostScript language output to put in the appropriate sequence of code to use this convention.

```
%!PS-Adobe-3.0
%%DocumentNeededResources: font Palatino-Roman
%%EndComments
%%EndProlog
%%Page: 1 1
%%BeginPageSetup
/pagelevel save def
%%EndPageSetup
%%IncludeResource: font Palatino-Roman
   /Palatino-Roman findfont 24 scalefont setfont
   100 100 moveto (Palatino Roman) show
pagelevel restore
showpage
```

5.1 Permanently Downloading Fonts

A permanently downloaded font is one that persists from one print job to another (although it usually will not persist if the machine is powered off). To permanently download a font, slightly different steps must be taken. It might not always be possible to accomplish this kind of downloading, depending on the environment and the device. In general, permanently downloading fonts is a function that a system administrator performs rather than an application.

The mechanism usually required to cause a font definition to persist in memory is to use the **exitserver** operator, which allows a program to exit the context that performs the **save** and **restore** between print jobs on a PostScript interpreter. (It is known as the "server loop.") Using **exitserver** is a system-level operation, and should never be performed by user-level print jobs. It requires a password to execute correctly.

Using **exitserver** requires a separate print job (which means an end-of-file must be sent afterwards). All memory consumed during an **exitserver** job is permanently in use until the interpreter is restarted, so use caution when using it. The following is the appropriate format for downloading a font program so it will remain in the printer's memory across all print jobs:

```
%!PS-Adobe-3.0 ExitServer
%%BeginExitServer 0
serverdict begin
   0 exitserver  % password is 0
%%EndExitServer
%%BeginResource:font: Optima
%!PS-Adobe Font-1.0 Optima 001.001
   15 dict begin
      /FontName Optima def
      % ...
   currentdict end
   dup /FontName get exch definefont pop
%%EndFont
%%EOF
```

The entire PostScript language font program is preceded by the call to **exitserver** as shown, and an appropriate end-of-file indication is sent afterward. The comments %%BeginExitServer and %%EndExitServer are necessary so that document spoolers can correctly catch the **exitserver** job and remove the code if necessary.

5.2 Disk-Based Fonts

In PostScript interpreter implementations that have a file system available, it is possible to place font programs on the disk in such a way that they can be used directly by the interpreter. Cartridge fonts also are a special case of the file system. The interpreter treats the ROM in a font cartridge as disk space, except that it is read-only.

In current PostScript interpreter implementations this works through an extended **findfont** operator. The **findfont** operator searches for a font given a key. It first looks in the dictionary **FontDirectory**. If the key is defined there, then the appropriate font dictionary is returned on the operand stack. If not, then **findfont** will look on the disk.

Font programs must be in a specific directory on the disk for **findfont** to locate them. In particular, the font program must be written into a file with a filename constructed as follows:

`fonts/`*FontName*

where *FontName* is the PostScript language name of the font as presented to **findfont**. For instance, if the font name is AvantGarde-BookOblique, the disk file name should be expressed as a PostScript language string, in this case

```
(fonts/AvantGarde-BookOblique)
```

The **findfont** operator will then **run** the file if it finds it (which has the effect of executing the contents of the file), and then attempt to **get** the font from the **FontDirectory**. The assumption is made that the effect of executing the file was an invocation of **definefont** appropriate for that font name.

The following is a PostScript language fragment that will cause a font program to be written to an appropriately-named disk file if it is pre-pended to the font program itself and downloaded (with an end-of-file afterward) to an interpreter with a disk file system:

```
%!
% writetodisk.ps

/fd (fonts/FontNameHere) (w) file def
/buff 128 string def
{ %loop
   currentfile buff readstring { %ifelse
      fd exch writestring
   }{ %else
      dup length 0 gt { %ifelse
         fd exch writestring
      }{ %else
         pop
      } ifelse
      fd closefile
      exit
   } ifelse
} bind loop
put the font program here
%%EOF
```

5.3 Adobe Disk-Based Font Downloader

Adobe has a Font Downloader product that downloads fonts to a printer's hard disk on request. This program does not use the previous algorithm. Instead, when the file (fonts/*FontName*) is run, a much smaller font dictionary is registered that has pointers to the actual character descriptions. This saves a great deal of memory in the PostScript interpreter, permitting more fonts to be used simultaneously.

It is important to note that the font files downloaded by this utility are not quite the same as, for example, putting the entire font program intact into the (fonts/*FontName*) file.

6 Font Naming

Fonts have many names. The important name from the PostScript interpreter's standpoint is the one by which it is registered with **definefont**. This is its PostScript language font name. A font program on the disk of a host system might reside in a file with a strange name, depending on the system.

These file naming conventions are important to a font downloading program, because they are the means by which a font must be located to download it.

6.1 Screen Font Names

On systems that use screen fonts, there must be a mechanism to associate the fonts used on the display with the fonts used during the printing process. The best way to accomplish this is to associate them with the PostScript language font names. For example, in the Macintosh screen font format there is a resource containing the correct PostScript font name for the font.

Given the correct PostScript language font name, the file name in which the downloadable font program itself is stored can usually be located by deriving the file name algorithmically from the PostScript language font name.

6.2 MS-DOS File Names

On an MS-DOS system, for instance, the file names are restricted to 8 characters, whereas the PostScript language font name can approach 30 characters in some instances. The scheme for determining (from the PostScript language font name) the disk file name can vary from one application to another. IBM is developing a standardized naming scheme that can be used by many vendors to minimize duplication of font files for different programs and environments.

6.3 Macintosh File Names

In the Macintosh environment, there is another naming scheme that is constructed from the PostScript language font name. A file name is constructed by dissecting the PostScript language font name into components based on capital letters and hyphens. Then the first five letters of the first name component are used, followed by the first three letters of any subsequent name components. The following are examples to illustrate this concept:

*Palat*ino-*Rom*an	=>	PalatRom
*Palat*ino-*Bol*d-*Ita*lic	=>	PalatBolIta
*Optim*a	=>	Optim
*Optim*a-*Bol*d-*Obl*ique	=>	OptimBolObl

7 Screen Fonts

For display purposes, Adobe Systems provides 72 and 96 dot-per-inch resolution screen font bitmaps that are matched to our PostScript language outline printer fonts. These screen fonts are generated from the printer fonts algorithmically, and hand-edited in some instances to accurately match the printer fonts. The fonts are distributed in several different formats.

- Macintosh FONT resource form.

- Microsoft Windows screen font format.

- BDF (Bitmap Distribution Format) As generic bitmap font files. These files are in a human-readable ASCII format devised by Adobe for distribution and interchange of bitmap fonts. They can be used in an environment in which Windows or Macintosh format screen fonts are not usable.

- ABF (Adobe™ Binary Format). These are generic bitmap files compressed into a much smaller format. At the moment, these are only available on 5-1/4” IBM PC diskettes.

In the Macintosh environment there is an associated resource known as the FOND resource, which contains a name mapping table that contains the correct PostScript language font names for each of the screen fonts, width information for each character, and kerning tables.

8 Font Character Width Information

Adobe Systems distributes font metric information (character widths, bounding boxes, encodings, ligature, and kerning tables) for each PostScript language font program. This information is distributed in the form of an Adobe Font Metrics (AFM) file. These files are human-readable ASCII text files in a line-oriented format intended to be parsed by software. In some environments, these files are parsed into more compact width tables, for example, the Macintosh FOND resource or the TeX "TFM" files. Font metric information is used by document composition software to lay out lines of text for proportionally spaced fonts.

The format of AFM files is described in the *Adobe Font Metrics File Format Specification*. The source for a sample AFM file parser, written in C language code, is included in the PostScript Language Software Development Kit.

Appendix: Changes Since Earlier Versions

Changes since May 4, 1991 version

- Document was reformatted in the new document layout and minor editorial changes were made.

Changes since January 16, 1989 version

- A few layout styles were changed, trademark information was cleaned up, and a few other minor modifications were made.

- The cover addresses and phone numbers were updated.

Index