New Technical Notes

Macintosh

Developer Support

PostScript Q&As

Imaging

Revised by: Developer Support Center May 1993
Written by: Developer Support Center October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

PostScript documentation

Date Written: 7/13/90
Last reviewed: 8/1/92

Adobe Systems publishes three great manuals that can help you with PostScript commands: *PostScript Language Reference Manual, PostScript Language Tutorial,* and *PostScript Language Cookbook.* You can find these manuals at just about any bookstore, or you can order them through APDA by calling (800) 282-2732 or writing them at:

Apple Computer, Inc.
20525 Mariani Avenue, Mail Stop 33-G
Cupertino, CA 95014

AppleTalk packets and PostScriptHandle PicComment

Date Written: 8/7/90
Last reviewed: 8/1/92

What's the optimal amount of PostScript to pass in PicComment 192 for AppleTalk packet throughput?

There is no one optimal size that your PostScriptHandle PicComment should be for AppleTalk packets. The LaserWriter driver sends different amounts of PostScript, depending on the

amount of memory that is available. It could send as little as 1K or as much as 4K bytes of PostScript at one time. The driver is using four 4096-byte buffers to send the data, and the amount of data that it sends from a particular buffer is completely memory dependent. In the general case, it does not send any data until a particular buffer is full. The driver does not control the number of packets that are sent over AppleTalk. When PAP receives the PostScript, it packages up the PostScript and sends it down to the LaserWriter. In general, DTS recommends that applications send a PostScript handle that contains 4096 bytes of data.

Using PostScript to control LaserWriter serial I/O
Date Written: 12/6/90
Last reviewed: 3/28/93

Where can I find information on using PostScript to control a multi-tray sheet feeder by communicating through the LaserWriter serial port?

—

You need to talk with Adobe on how they control the serial chip with PostScript. They use some undocumented PostScript calls to control the serial chip on the controller board of the LaserWriter. They can be reached at:

Adobe Systems, Inc.
1585 Charleston Road
Mountain View, CA 94039

or

AppleLink Address: ADOBE.SUPT (Tech Support)

Details about installing feeder resources for Macintosh into the LaserWriter driver 7.0 dialogs are available in the Macintosh Technical Note “Feeder Fodder.”

How to inhibit LaserWriter test page
Date Written: 12/24/90
Last reviewed: 8/1/92

How do I inhibit the test page on a LaserWriter?

—

1. Use the LaserWriter Utility to turn the startup page off (best), or

2. Download the following PostScript commands:

```
serverdict begin 0 exitserver
```

```
statusdict begin
false setdostartpage
end
```

The above commands can be downloaded from your Macintosh to the printer several ways:

1. Use the menu command in the LaserWriter Utility that comes with System 7 (easiest).
2. You can use a PostScript utility like LaserTalk from Emerald City Software.

3. The PostScript Language facility in Microsoft Word.
4. PSDump or PSsend
5. Use the PostScript utilities in the Developer CD's PostScript Utilities folder.
6. Use the methods described in the Macintosh Technical Note #91.

When you issue the commands, your LaserWriter will probably crank and grind for a few seconds (it has to re-program the EPROM inside) then after the lights stop flashing, your LaserWriter will no longer spit out the startup page until you enable the setdostartpage flag again (in other words, this isn't a permanent change).

For more information on the above subject we recommend the *PostScript Language Reference Manual* by Adobe Systems.

Where to find Encapsulated PostScript file (EPSF) documentation

Date Written: 4/5/91
Last reviewed: 8/1/92

I'm developing an application that creates pictures containing both QuickDraw and PostScript (as picComments). How do I create an encapsulated PostScript file (EPSF)? Is the format explained anywhere, and is it possible to convert the QuickDraw into PostScript directly in the application?

—

Adobe's EPSF format is documented in the new (2nd) edition of Adobe's *PostScript Language Reference Manual* (Addison-Wesley) and in "Adobe Document Structuring Conventions," available from Adobe Systems. Adobe's references explain how to structure your comments in your PostScript file so that other applications know how to read it.

You probably don't want to try to convert QuickDraw to PostScript directly in your application, although it is possible. Basically, it involves translating each QuickDraw command into PostScript, which is what the LaserWriter Driver does. It's a lot of work and makes you dependent on the QuickDraw features being used, which can be a problem when new features are added.

PostScript setprintername doesn't accept colons in string

Date Written: 4/16/91
Last reviewed: 8/1/92

Why can't I use a colon with the PostScript command setprintername to change the Personal LaserWriter NT network name and type, but it works with the LaserWriter II?

—

The reason that the Personal LaserWriter NT is not accepting the PostScript you're sending is not because of a bug, but rather a bug fix. Colons have never been acceptable in the string passed to setprintername. (See page 296 of Adobe's *PostScript Language Reference Manual.)* The reason the call works in older LaserWriters but not the Personal LaserWriter NT is that the NT has a newer version of PostScript in it. Older versions of PostScript didn't enforce Adobe's printer naming conventions. The setprintername restrictions have been enforced in more recent versions, however. Therefore, colons are not allowed.

If you need to change the LaserWriter network type, it can be done with the AppleTalk-type parameter described on page 105 of Apple's *LaserWriter Reference* manual. This method will work on the new printers as well as the old.

PostScript code for renaming LaserWriter

Date Written: 6/5/91
Last reviewed: 8/1/92

In checking a printer name to ensure an even number of characters, does the LaserWriter Namer append a space to the end if the total is odd?

—

In a nutshell, yes. The Namer does indeed append an extra space to odd-length printer names. The reasoning behind this is to be compatible with some of the old 68000 processors. The addressing scheme of these old processors requires that memory be accessed by even-numbered addresses. To get at a piece of memory located after an odd-length LaserWriter name, you would need to get an odd-numbered address (which is not directly possible). This obviously can be worked around, but the authors of the Namer felt it best to avoid this altogether by padding the name when necessary. It's worth noting that the version 7.0 Namer does this too.

If you'd like to work around this, here's an alternate method for renaming the printer:

The printer name is stored as a string in the persistent parameters of all LaserWriters. This string can be changed by the following PostScript program:

```
serverdict begin 0 exitserver
statusdict begin (NEWNAME) setprintername
```

Replace NEWNAME with the name you want for the printer. The name can be up to 31 characters and should consist entirely of printing characters--the "@" and ":" cannot be used. Note that parentheses are required.

This program can be downloaded to the LaserWriter in batch or interactive mode. In interactive mode you should type both command strings on the same line before pressing Enter, because exitserver disconnects you from the printer. When you reestablish your connection to the printer, the name will be changed.

Make sure you have selected the correct LaserWriter using the Chooser before downloading this program, so you don't rename the wrong LaserWriter.

Printing 2-byte bitmap fonts on PostScript LaserWriters

Date Written: 9/29/91
Last reviewed: 8/1/92

The Macintosh Technical Note #91 refers to using QuickDraw picComments to print rotated text, but this doesn't work for 2-byte Japanese text. Is there a recommended workaround for this problem?

—

Two-byte characters don't print when using the TextBegin, TextCenter, and TextEnd PicComments because of the structure of the data that is being sent in the comments. The LaserWriter driver uses these PicComments to rotate text, and it is not expecting the data to be in the 2-byte format. The 2-byte format is converted to bitmap data by the driver, so the data is ignored and nothing is printed. The driver is expecting the comments to contain "normal" text.

Since the data is in bitmap format, you can use the RotateBegin and RotateEnd PicComments to perform the rotation, as follows:

```
RotateBegin
   DrawString (...);  << To rotate your 2 byte Japanese characters
RotateEnd
```

The tricky part of the solution is that you only need to use this approach when you're printing 2-byte Japanese bitmap fonts. Therefore, you'll need to check the type of data that is to be printed, and send the appropriate PicComments.

You can also use the following PicComments to hide the QuickDraw representation of the rotated text, as shown below:

```
PostScriptBegin
   CopyBits (...);   << For the QuickDraw printers
PostScriptEnd
```

The PostScriptBegin and PostScriptEnd PicComments are used to hide the CopyBits calls from the LaserWriter driver in a device-independent manner. When you use PostScriptBegin, the LaserWriter driver doesn't interpret any QuickDraw calls until it finds the PostScriptEnd PicComment. Since QuickDraw printer drivers do *not* understand the PostScriptBegin and PostScriptEnd PicComments, they ignore the PicComments, and the CopyBits call is used to perform the rotation of your text.

There is one problem with the previously mentioned solution: If you record the previous PicComments into a picture and print it, everything will print exactly as expected, but if you display the picture into a window, you'll see the *unrotated* version of the text. This is definitely a bug, which has been reported, but it won't be fixed in the very near future. So, what do you do in the meantime? You have the following two options:

1) Do not export pictures that contain this type of rotated data.
2) Export the picture with this problem, and explain the problem to your users.

Neither solution is ideal. Quite a few applications out in the market today have decided to use idea #2.

Macintosh spooling should be device independent

Date Written: 1/1/90
Last reviewed: 8/1/92

How can I tell whether the chosen printer supports PostScript or not?

—

There is no supported method for determining whether or not a device speaks PostScript. Apple engineering would like all Macintosh applications to spool both PostScript and

QuickDraw regardless of the chosen printer, so that the spool file is device-independent and can be redirected after spooling. Since engineering is strongly encouraging developers to spool both models, it is doubtful that Apple will be providing a query mechanism in the future. The best method is to assume PostScript is available, and format your document based on information returned by PrGeneral and the print record.

LaserWriter driver and PostScript showpage

Date Written: 1/1/90
Last reviewed: 8/1/92

When I print a PostScript file, why do I get one blank page for every page printed?

—

Your PostScript file probably contains calls to the PostScript showpage operator. This operator is also called by the Macintosh LaserWriter driver when it receives the PrClosePage call. The solution to this problem is to strip showpage calls from the file before sending the file to the LaserWriter. If you send one QuickDraw graphic at the beginning of each page,

```
MoveTo(-32000, -32000);
Line(0, 0);
```

the LaserWriter driver will handle calling showpage for you. Versions of the LaserWriter driver before 5.2 do not send the showpage operator unless a QuickDraw drawing routine (that is, something other than PicComment) is sent. This is why you need to do the MoveTo/LineTo. Versions 5.2 and later will send the showpage operator whether any QuickDraw has been sent or not. Since you can't be sure whether showpage will be sent for you or not, the best solution is to let the LaserWriter driver send it.

PicComment TextIsPostScript versus PostScriptHandle

Date Written: 1/1/90
Last reviewed: 8/1/92

When using the Macintosh TextIsPostScript picture comment, my PostScript code is also "printed" on the ImageWriter. Why isn't it ignored?

—

One of the comments, TextIsPostScript, allows the application to use the QuickDraw DrawString procedure to send the PostScript code. This method is no longer recommended, as it is not device independent. If you use the TextIsPostScript comment when printing to a non-PostScript device, the PostScript code will be printed as text. On an ImageWriter, the entire PostScript program will be printed as one line on the page, because the TextIsPostScript PicComment is ignored by the ImageWriter drivers, and the DrawString calls are executed normally.

To avoid this problem, developers should use the PostScriptHandle comment. This comment accepts a handle as a parameter. The handle points to the PostScript code that you want to download. This comment is better for two reasons:

First, there is no QuickDraw/PostScript interaction. For example, early versions of the

LaserWriter driver sent the strings passed to DrawString as-is. The latest version of the driver

appends carriage returns onto those strings. If your application is expecting the DrawString calls to work one way or another, you might get a surprise.

The other reason for using the PostScriptHandle comment involves device independence. When a non-PostScript driver sees the PostScriptHandle PicComment, it not only ignores the comment, but also ignores the PostScript data that was passed to the comment, so nothing prints if you use the PostScriptHandle comment on a non-PostScript driver.

X-Ref:
Macintosh Technical Note "How To Produce Continuous Sound Without Clicking"

How to send PostScript from the Macintosh to the LaserWriter
Date Written: 5/3/89
Last reviewed: 8/1/92

How do I send PostScript from the Macintosh to the LaserWriter?

—

The best method for sending PostScript to the LaserWriter is to use the PostScriptHandle (kind = 192) PicComment documented in the Macintosh Technical Note "Optimizing for the LaserWriter—Picture Comments." There are a few PicComments for sending PostScript to the LaserWriter, but the easiest to use and most problem free is the PostScriptHandle comment. One model is:

```
PrOpenPage(...)
{ Send some QuickDraw so that the Printing Manager gets a  }
{ chance to define the clip region.             }
MoveTo(-32000, -32000);
Line(0, 0);

PicComment(PostScriptBegin, 0, NIL);

{ QuickDraw representation of graphic. }
MoveTo(100, 100);
LineTo(200, 200);

{ PostScript representation of graphic. }
thePSHandle^^ := '100 100 moveto 200 200 lineto stroke';
PicComment(PostScriptHandle,GetHandleSize(thePSHandle),thePSHandle);

PicComment(PostScriptEnd, 0, NIL);
PrClosePage(...)
```

The described fragment prints a line on any type of printer, PostScript or not. The first MoveTo/Line combination is required to give the LaserWriter driver a chance to define a clipping region. The LaserWriter driver replaces the grafProcs record in the GrafPort returned from PrOpenDoc. In order for the LaserWriter driver to get execution time, you must execute a QuickDraw drawing routine that calls one of the grafProcs. In this case, the MoveTo/Line combination is sent firing the StdLine grafProc. When StdLine executes, it notices that the GrafPort has been reinitialized, and therefore initializes the clipping region for the port. Until the MoveTo/Line combination is executed, the clipping region for the port is set to (0,0,0,0). If PostScript code is sent via the PostScriptHandle PicComment before executing any QuickDraw routines, all PostScript operations will be clipped to (0,0,0,0). The next thing to do is send the PostScriptBegin PicComment. This comment is recognized only by PostScript printer drivers.

When the driver receives this comment, it saves the current state of the PostScript device (by executing the PostScript gsave operator), then disables all QuickDraw drawing operations. This way, the QuickDraw representation of the Graphic will be ignored by PostScript devices. In the previous example, the MoveTo/LineTo combination is only executed on non-PostScript devices. The next comment is the PostScriptHandle PicComment. This tells the driver that the data in thePSHandle is to be sent to the device as PostScript code. The driver then passes this code unchanged to the PostScript device for execution. The PostScriptHandle PicComment is only recognized by PostScript printer drivers. The last PicComment, PostScriptEnd, tells the driver to restore the previous state of the device (via a PostScript grestore call), and to enable QuickDraw drawing operations. Since all PicComments are ignored by QuickDraw devices, only the QuickDraw representation is printed. Since PostScriptBegin tells PostScript drivers to ignore QuickDraw operations, only the PostScript representation is printed on PostScript devices. This is a truly device-independent method for providing both PostScript and QuickDraw representations of a document.

Converting PostScript file to Encapsulated PostScript

Date Written: 5/19/92
Last reviewed: 8/1/92

What needs to be added to an ordinary PostScript file to convert it to an Encapsulated PostScript (EPS) file?

—

The latest Adobe “red book,” PostScript Language Reference Manual (2nd Edition), has the information you need. Appendix H: Encapsulated PostScript File Format Version 3.0 describes the contents of an EPS file. Appendix G: Document Structuring Conventions specifies the single-page document PS file, which must include at least these two comments:

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: llx lly urx ury
```

You might find the following references from Adobe useful as well:

Adobe Illustrator Document Format Specification v2.0 (Adobe Technical Note #LPS5007)
Encapsulated PostScript Files v1.2 (Adobe Technical Note #LPS5002)

You can reach Adobe at:

Corporate Headquarters
1585 Charleston Road, PO Box 7900
Mountain View, CA 94039-7900
(415) 961-4111 Developer Support

Macintosh Technical Note “Picture Comments—The Real Deal” (formerly #91) contains sample code for using all of the supported Picture Comments.

Printing halftone images on the LaserWriter

Date Written: 5/3/89
Last reviewed: 8/1/92

How can I print halftone images on the LaserWriter? How do I use the PostScript “image” operator for printing halftones?

—

The PostScript “image” operator is similar to the Macintosh QuickDraw CopyBits routine. It can be used by converting a QuickDraw bitmap/pixmap into an image call. This is only necessary if you are printing color information with a PostScript printer driver that doesn’t support Color QuickDraw. LaserWriter drivers up to 5.2 do NOT support Color QuickDraw. If you want to send color information to a LaserWriter with version 5.2 of the driver, you will have to convert your color images into image commands. Before doing this it is important to check the GrafPort returned by PrOpenDoc. If this port is a Color QuickDraw port (for example, portBits.rowBytes < 0) then you should use Color QuickDraw commands and let the driver handle the conversion to PostScript.

About image: The PostScript image operator is used to send bitmaps or pixmaps to the LaserWriter. The image operator can handle depths from one to eight bits per pixel. Our current LaserWriters can image only about 16 shades of gray, but the printed page will look like there’s more. Because the image operator is still a PostScript operator, it expects its data in the form of hexadecimal bytes. The bytes are represented by two ASCII characters(“0”-”9,” “A”-”F”) per byte. The image operator takes these parameters: width height depth matrix image-data

The first three are the width, height, and depth of the image, and the matrix is the transformation matrix to be applied to the current matrix. See the PostScript Language Reference Manual for more information. The image data is where the actual hex data should go. Instead of inserting the data between the first parameters and the image operator itself, it is better to use a small PostScript procedure to read the data starting right after the image operator. For example:

```
640 480 8 [640 0 0 480 0 0] {currentfile picstr readhexstring pop} image FF 00
FF 00 FF 00 FF 00...
```

In the preceding example, the width of the image is 640, the height is 480, and the depth is 8. The matrix (enclosed in brackets) is set up to draw the image starting at QuickDraw’s 0,0(top left of page), and with no scaling. The PostScript code (enclosed in braces) is not executed. Instead, it is passed to the image operator, and the image operator calls it repeatedly until it has enough data to draw the image. In this case, it is expecting 640*480 bytes. When the image operator calls the procedure, it does the following:

1. Pushes the current file which in this case is the stream of data coming to the LaserWriter over AppleTalk. This is the first parameter to the readhexstring operator.

2. Pushes picstr. picstr is a string variable defined to hold one row of hex data. The PostScript to create the picstr is:

   ```
   /picstr 640 def
   ```

3. readhexstring is called to fill picstr with data from the current file. It begins reading bytes which are the characters following the image operator.

4. readhexstring leaves the string we want, and a boolean that we don’t want, on the stack, so we do one pop to kill off the boolean. Now the string is left behind for the image operator to use.

Using the above PostScript code you can easily print an image. Just fill in the width height and depth, and send the hex data immediately following the PostScript code.

Setting up for image: Most of the users of this technique are going to want to print a Color QuickDraw pixmap. Although the image command does a lot of the work for you, there are still a couple of tricks for performance:

Assume the maximum depth: Since the current version of the image operator has a maximum depth of eight bits/pixel, it is wise to convert the source image to the same depth before imaging. You can do this very simply by using an offscreen GrafPort that is set to eight bits/pixel, and then using CopyBits to do the depth conversion for you. This does a nice job of converting lower resolution images to 8 bits/pixel.

Build a color table: An eight bit deep image can use only 256 colors. Since the image that you are starting with is probably color, and the image you get will be grayscale, you need to convert the colors in the source color table into PostScript grayscale values. This is actually easy to do using the Color Manager. First create a table that can hold 512 bytes. This is two bytes for each color value from 0 to 255. Since PostScript wants the values in ASCII, you need two characters for each pixel. Now loop through the colors in the color table. Call Index2Color to get the real RGB color for that index, and then call RGB2HSL to convert the RGB color into a luminance value. This value is expressed as a SmallFract which can then be scaled into a value from 0 to 255. This value should then be converted to ASCII, and stored at the appropriate location in the table. When you are done, you should be able to use a pixel value as an index into your table of PostScript color values. For each pixel in the image, send two characters to the LaserWriter.

Sending the data: Once you have set up the color table, all that's left to do is loop through all of the pixels, and send their PostScript representation to the LaserWriter. There are a couple of ways to do this. One is to use the low-level Print Manager interface and stream the PostScript using the stdBuf PrCtlCall. Although this seems like it would be the fastest way, the latest version of the LaserWriter driver (5.0) converts all low-level calls to their high level equivalent before executing them, so the low-level interface is no longer faster than the high level. Another way is to use the high-level Print Manager interface, and send the data via the PostScriptHandle PicComment. This enables you to buffer a large amount of data before actually sending it. Using this second technique, you should be able to image a Macintosh II screen in about 5 minutes on a LaserWriter Plus, and about 1.5 minutes on a LaserWriter II NTX.

X-Ref:
PostScript Language Reference Manual, Adobe Systems.