Adobe Developers Association

ADA news

In This Issue

Volume 3, Number 7

Page Size Support for the Full Bodied Application

While the PostScript™ language has supported large format output for more than six years, the number of large format PostScript output devices on the market has increased of late. This article will discuss how application developers may better support the full range of page sizes now available for output.

As a rule, your application should offer its users the choice of every page size offered by a particular device, up to the maximum custom page size supported. Accomplishing this task is fairly straightforward, whether your application has its own PostScript printer driver or takes advantage of the standard PostScript printer drivers for the Windows™ and Macintosh® platforms. Three guidelines for your application are as follows: 1) Your application should put no unnecessary restrictions on the page size that a user can select; 2) Your application should display and support all of the printer's available page sizes; and 3) Your application should support custom page sizes.

Guideline 1: Do not impose unnecessary restrictions on your users' page size choices. Microsoft® Word version 5.0 on the Macintosh and Windows platforms imposes a maximum page height and width restriction of 22 inches. Similarly, Adobe™ Illustrator™ versions 3.2 and earlier impose a maximum artwork height and width of 18 inches. A workaround to this restriction is for the user to enlarge his/her output via the standard PostScript printer drivers on the Macintosh and Windows platforms. This is done via the reduce/enlarge option in the Page Setup dialog box. Setting the enlargement value to its maximum value of 400% increases the maximum output size in Microsoft Word to 88 inches square, and similarly in Adobe Illustrator, the artwork size may increase to 72 inches square. While this size output may seem sufficiently large, there are now PostScript output devices shipping which can output much larger page sizes. For example, one such plotter supports a page size referred to in its PostScript Printer Description (PPD) language file as "Lsize". Lsize paper is 34 inches high by 247.5 inches long. This plotter is just one of many large format PostScript output devices that have begun to ship recently. Figure 1 shows a small subset of the various paper sizes supported by PostScript output devices that are currently out on the market. Please see the column entitled "Large Format Devices" in this issue of the *ADA News* for a partial list of large format output devices that are now shipping.

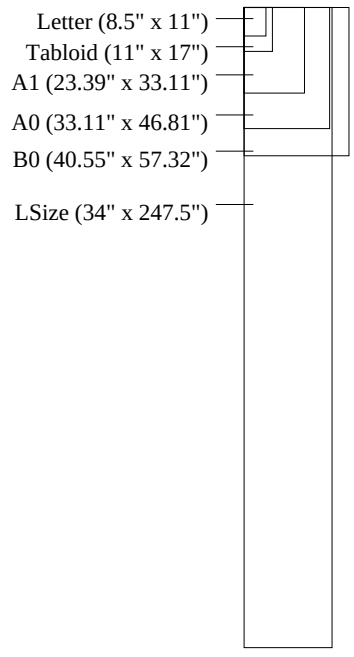

Figure 1. Standard and large format paper sizes.

continued on page 2

Adobe

How To Reach Us

DEVELOPERS ASSOCIATION HOTLINE:

U.S. and Canada:
(415) 961-4111
M–F, 8 a.m.–5 p.m., PDT.
If all engineers are unavailable, please leave a detailed message with your developer number, name, and telephone number, and we will get back to you within 24 hours.

Europe:
+31-20-6511-355

FAX:

U.S. and Canada:
(415) 967-9231
Attention:
Adobe Developers Association

Europe:
+31-20-6511-313
Attention:
Adobe Developers Association

EMAIL:

U.S.
devsup-person@mv.us.adobe.com

Europe:
eurosupport@adobe.com

MAIL:

U.S. and Canada:
Adobe Developers Association
Adobe Systems Incorporated
1585 Charleston Road
P.O. Box 7900
Mt. View, CA 94039-7900

Europe:
Adobe Developers Association
Adobe Systems Europe B.V.
Europlaza
Hoogoorddreef 54a
1101 BE Amsterdam Z.O.
The Netherlands

Send all inquiries, letters and address changes to the appropriate address above.

Page Size Support

To allow your users to take full advantage of their output device's range of page sizes, your applications should not arbitrarily hard code any maximum or minimum height or width values into their page size choices. Furthermore, in other areas of your application, do not assume that the page size will fall within a restricted range. This will affect the rulers you display, the zoom values you allow, the restrictions you place on type sizes, how you track internal document coordinates, and how you map your application's internal coordinates to the platform's drawing coordinates, among other areas.

Guideline 2: Display and support all of the printer's page size choices.
Most output devices support several distinct page sizes. Below we outline what is involved in supporting all these page size options from within your application.

Supporting Page Size Options on the Windows Platform
Applications that use the standard PostScript printer driver on the Windows platform may display all page sizes for the user with a few basic calls. An application normally calls the `PrintDlg( )` function to display a Print dialog box or a Print Setup dialog box. The Print dialog box makes it possible for the user to specify the properties of a particular print job, such as page range, resolution, and number of copies. The Print Setup dialog box makes it possible for the user to configure the printer and select document properties, such as page size, desired printer, paper source and page orientation. The application calls `PrintDlg( )` with a pointer to a `PRINTDLG` structure that contains information used to initialize the dialog box. When the `PrintDlg( )` function returns, the `PRINTDLG` structure contains information about the user's selections. By this means, Windows environment's graphics device interface (GDI) displays all page sizes choices for a given device, and then returns the user's selected page size to the application via the `PRINTDLG` structure.

There are two limitations that may restrict the page sizes available to GDI applications. The first limitation is imposed by the structure that GDI uses to communicate with the printer driver, and the second is imposed by the structure that an application uses to communicate with GDI and the printer driver.

The GDIINFO structure
`GDIINFO` is a private structure, defined by Microsoft, by which GDI transfers information to and from the printer driver. The `GDIINFO` structure contains fields for the printer's resolution, and selected paper height and paper width, among other values.

A PostScript language driver provides page size information to GDI through `GDIINFO`, in terms of output device pixels. So, for example, if the user has selected a resolution of 400 dots per inch (dpi) for output, then an 11 inch paper height would be specified as "4,400 pixels". For an imagesetter, a 17 inch paper height at an output resolution of 1200 dpi would be specified as "20,400 pixels." One can quickly see that this page size value may be quite large, for a given

paper size and resolution combination. Unfortunately, this value must fit into a signed, 16-bit integer in the GDIINFO structure. To work around this limitation, Adobe's PostScript printer driver for Windows will reduce the pixel resolution value that it passes to GDI, if necessary, to specify the current page size. If the product of the output resolution and one of the page dimensions is greater than 32K, the driver will divide the resolution value by two. The driver will continue to halve the resolution value that it passes to GDI until the page size values can fit into a signed, 16-bit integer. For applications that rely on GDI calls for their output, this potential reduction in resolution may affect the accuracy of object placement, but will not affect the output quality. The printer driver sets the printer to the user's requested resolution, regardless of the value passed back to GDI. GDI applications should avoid making any calls that rely upon the output resolution value; keep your page descriptions both device and resolution independent. Applications that generate their own PostScript language code and pass that through the standard driver will not be affected by this resolution reduction.

The DEVMODE structure

Windows applications may provide a DEVMODE structure to PrintDlg( ), to obtain the apparent printer resolution and current paper selection, among other information. Furthermore, the application can call the driver directly to obtain paper size information. It does this by calling the DeviceCapabilities( ) function with a parameter specifying what information it needs; for example, the application would pass a value of DC_PAPERNAMES, DC_PAPERS, or DC_PAPERSIZES as the nIndex parameter to obtain paper size information. Paper sizes may be one of a set of Microsoft's defaults. If the user chooses one of the default values, then the dmPaperSize field of the DEVMODE structure will contain one of these pre-defined choices, such as DMPAPER_LETTER, DMPAPER_A3, or DMPAPER_11X17. Alternatively, if the user selects a page size that is not in Microsoft's default set, the DEVMODE will contain dmPaperLength and dmPaperWidth fields that are set to specify the height and width of the selected page size. These paper size values are specified in terms of tenths of millimeters; and, since these values must be stored in signed, 16-bit integers, the maximum page size is limited to 129 inches square. There is no workaround for this limitation, for GDI applications. Applications that display their own Page Setup and Print dialog boxes, and generate their own PostScript language code to pass through the standard driver, will not be affected by this 129 inch limit.

The DeviceCapabilities( ) function and the DEVMODE structure are documented in the Windows 3.1 Software Development Kit (SDK).

Supporting Page Size Options on the Macintosh Platform

Applications that use the standard PostScript printer driver on the Macintosh platform may display all page sizes for the user by means of a few basic calls. Macintosh applications generally call PrOpen( ) and PrintDefault( ) with the handle to a previously-allocated print record block. PrOpen( ) initializes the Print Manager and opens the printing resource file, and

continued on page 4

L A R G E Format Devices

A number of printer manufacturers have begun shipping PostScript language output devices, which support very large paper sizes. Some current uses for such devices are as follows:

- Large format output is being laminated and tiled to be used for signs on the side of trucks. Similarly, large strips of PostScript output are tiled together and pasted on to billboards.
- CAD and architectural drawings.
- Printing of maps.
- Grocery stores, department stores and large boutiques are using this technology to output store banners for sales and special events.
- In addition to these uses, large format output is now available for PostScript digital photography, enabling production of poster-size enlargements of images.

Here is a partial list of some of the large format devices currently out on the market. The devices all support PostScript language Level 2, unless otherwise noted.

Page Size Support

PrintDefault( ) causes the driver to fill the print record block with default information for the chosen output device. When a user selects Page Setup, the application calls PrOpen( ), and then calls PrStlDialog( ) with a handle to the print record. During PrStlDialog( ), the Page Setup dialog is displayed with all the page size options available for the selected device. When the user selects the desired page size, that choice is entered into the print record for the current document. The application may directly obtain this page size information by checking the rPaper and rPage values of the print record block. rPaper, which indicates the dimensions of the physical paper size, is a Rect field of the TPrint structure. rPage, which indicates the imageable area of the page, is a Rect field of the TPrint structure's TPrinfo field, prinfo. The values stored in rPage and rPaper indicate the page dimensions in terms of the current resolution. The default resolution is 72 dpi, but applications that draw their page output using QuickDraw™ calls often change this resolution to 288 dpi or 300 dpi, using the PrGeneral call with the SetRsl opcode.

As in the Windows programming environment, QuickDraw's rPage and rPaper values are stored in signed, 16-bit integers. However, changing the resolution will affect both output quality and accuracy of placement for applications that rely on QuickDraw calls for their page descriptions. Applications that generate their own PostScript language code, passing it through the driver, are unaffected by the QuickDraw resolution.

If a QuickDraw application designates a resolution of 2400 dpi, in order to print to an imagesetter, that will limit the largest page dimension to 13.65 inches, since the product of resolution and the largest page dimension must be stored in a signed, 16-bit integer. There is no workaround for this limitation for applications that are describing their output using QuickDraw calls; these applications may cautiously increase the resolution to allow for better object positioning and better quality output. If your application outputs its own PostScript language code, using the standard PostScript printer driver only to pass the code to the printer, your application should *not* call PrGeneral( ) to change the resolution value. This will leave the resolution at its default value of 72 dpi, allowing for page dimensions of up to 455 inches. If your application is passing its own PostScript language code through Adobe's PostScript printer driver, PSPrinter 8.x, then it may reduce the resolution to 25 dpi; be aware that other drivers may not support this low resolution value.

For details on the above mentioned data structures and the Macintosh Print Manager routines, please refer to *Inside Macintosh,* Volume II, Chapter 5.

Supporting Page Size Options in Your Own Driver

If your application is on the DOS or UNIX® platform or has its own PostScript printer driver under the Macintosh or Windows environment, your driver can obtain page size information by parsing the printer's PostScript Printer Description (PPD)language file. Specifically, a *PageSize and *PageRegion keyword will be present for every page size choice available on the device and will give invocation code for selecting that page size. The *PageDimension entries

Large Format Devices (continued)

3M ScotchPrint Server
Driving Rastergraphics
Maximum Width 22" to 24"

Driving Synergy Plotter
Maximum Width 36"

Driving HP DesignJet 650c
Color Inkjet Plotter
Maximum Width 36"

Cactus® 75/101
Driving Versatec® 8900™
series of plotters
Widths of 36", 44", and 54"

Colossal Graphics Power RIP
Driving HP DesignJet 650c
Color Inkjet Plotter
Maximum Width 36"

Hewlett Packard
DesignJet 650c
Color Inkjet Plotter
Maximum Width 36"

Management Graphics Inc. JetStream Color Image Server
Driving Versatec 8900 series
Widths of 36", 44", and 54"

Xerox Engineering Systems XES Level 1 PostScript Converter for Serveware
Driving Versatec 8900
series of plotters
Widths of 24", 36", and 44"

Page Size Support

indicate the size of each page and the *ImageableArea entries indicate the printable region of each page. Your PostScript printer driver should parse the PPD file for this information and display all page size choices to the user. For more details about the syntax and placement of these PPD keywords, see Technical Note #5003, "PostScript Printer Description File Format, version 4.2." Sample code for parsing PPD files is available in our PostScript Language Software Development Kit.

Guideline 3: Support Custom Page Sizes

Roll-fed devices such as imagesetters or plotters can support a myriad of custom pages sizes, since the paper or film may be cut in an infinite number of places. Similarly, cut-sheet devices support printing on pre-cut paper of any size. By supporting custom page sizes within your application, you allow your users to select the optimum page size for his or her needs. On a device that can support up to a 200 inch page length, it would make little sense for a PPD file to list all possible page lengths, incrementing by a few inches at a time. Instead, the PPD indicates the valid ranges of page size values, enabling applications to specify a custom page size when appropriate.

Adobe's PostScript printer driver for Windows, versions 2.1 and later, automatically supports custom page sizes, if the selected printer supports them. Future versions of Adobe's PostScript printer driver for the Macintosh will support custom page sizes as well. With custom page size support, a "Custom Paper" or "Custom" entry simply shows up as one of the page size choices your users can select. After selecting the custom page size, your user can indicate the desired height and width for output. Your application does not need to be modified to support custom page sizes, but you may wish to make some of the changes suggested in the "Guideline 1" section of this article.

If you have your own PostScript printer driver, you can support custom page sizes by adding "Custom" to the list of possible page sizes for the user. If the user selects Custom, your driver should display an interface, allowing the user to choose his/her desired page height and width. In the selected printer's PPD file, your application should parse for the keyword *CustomPageSize which will be followed by invocation code for specifying a custom page on that device. If the *CustomPageSize keyword does not appear in the PPD, then the selected device does not support custom page sizes, and your application should not activate the Custom page option for the user. If the keyword is present, it will be accompanied by the *ParamCustomPageSize keywords, indicating the ranges of valid page width, height, width offset, height offset and orientation values that are valid. This information will allow your application to do bounds checking on the values entered by your users, and inform the users of these limits. The information in the PPD will also provide you with the parameters needed to display the proper imageable area for a custom page. Details about these keywords are documented in Technical Note #5003, "PostScript Printer Description File Format, version 4.2. "

continued on page 6

Page Size Support

Summary

Now that an increasing number of PostScript output devices are emerging which support very large paper sizes, application developers should take another look at their user-interfaces and make sure they aren't imposing unnecessary restrictions on their user's output choices. Five years ago we may have thought, "there will never be an output device that supports a 35 inch by 200 inch page." Well, that time has come. Our engineers have now redesigned the Adobe Illustrator application to support larger page sizes. Adobe Illustrator 5.0 for the Macintosh supports a maximum page size of 120 inches square, which allows for a 480 inch square page with the aid of driver enlargement. As you rethink your application's page size support, keep tomorrow's printers in mind. PostScript printers may be supporting even larger page sizes in the not too distant future. §

You Want Your Macintosh Application to be Able to Print!

So you're developing this great Macintosh application that will allow your users to do all these really cool things and you're starting to think about your print module. Where should you start? This article lists documents with Macintosh printing guidelines, presents some things you should know before you start, and points you to the relevant places for more information.

Follow the guidelines

Well, first you need to follow the rules and guidelines Apple® has set down; you need to know what Apple says about printing on the Macintosh. Apple provides a multitude of technical notes, Q&As and books which may be a little overwhelming to take in all at once. Adobe also provides some technical notes that are specific to drivers and the PostScript language. It can be difficult to figure out where to get all the information necessary to get started. Below is a summary of the literature you should familiarize yourself with to be able to proceed quickly with the development of your Macintosh application's print module.

Before you read any technical notes, be sure to read "The Printing Manager" sections of *Inside Macintosh*, Volumes II and V. These sections describe the foundation on which Apple's current printing architecture is based. They also include an overview of the Macintosh print loop, a description of the standard Macintosh print records and dialogs and a list of all the print manager routines.

Apple Technical Notes

Technical notes from Apple's Developer Technical Support group offer tips, tricks and certain truths about what is expected of the application and the Macintosh operating system. They usually clarify misconceptions that developers have about Macintosh programming. You should read the following printing and PostScript language related technical notes from Apple:

- **PR 10** (previously #161):

"A Printing Loop That Cares" describes when and how an application should open and close the print manager.

- **PR 02** (previously #122):

"Device-Independent Printing" describes the importance of not relying on the current structure of the print record.

- **PR 04** (previously #72):

"LaserWriter® Optimization Techniques" describes the limitations of LaserWriter printers and ways to ensure efficient printing on them. LaserWriter printers are PostScript output devices and many of the tips in this technical note apply to PostScript printing in general.

continued on page 8

Macintosh Application Printing

• **QD 10** (previously #91):

"Picture Comments-The Real Deal" describes the picture comments supported by the LaserWriter 7.x printer driver and the Adobe PostScript printer driver, PSPrinter 8.x (a.k.a Apple LaserWriter 8.x).

• **PR 7** (previously #128):

"PrGeneral" describes an extension to the Printing Manager architecture as described in *Inside Macintosh*. The PrGeneral( ) procedure allows the addition of new functionality in LaserWriter drivers and is present in versions 4.0 and later.

You can obtain *Inside Macintosh* from any computer bookstore; you can get the Apple technical notes from Apple Programmers and Developers Association (APDA) or on Applelink: once you get to "Applelink Services: Support: Developer Support: Dev Tech Answers", enter the name of the technical note. For Macintosh technical support, contact Apple's Developer Support Center. They can be reached at (408) 974-4897.

Adobe Technical Notes

Technical notes from the Adobe Developers Association focus on aiding developers in taking advantage of PostScript language technology. The Macintosh driver-specific technical notes are:

• #5133: "Tips for Writing a Print Spooler" describes the details of specific queries issued by PSPrinter 8.x and the results it expects from the receiver of these queries, whether it be a PostScript interpreter or a print spooler.

• #5134: "Tips for Working with the New PostScript Printer Driver for Macintosh" gives tips on ensuring compatibility between Macintosh applications and PSPrinter and describes three new PrGeneral( ) selectors provided in PSPrinter versions 8.0 and beyond.

• #5139: "PSPrinter's PrGeneral Interface to Get the Communications Resource" describes a new PrGeneral( ) selector provided in PSPrinter versions 8.1 and beyond.

You can obtain these technical notes from the Macintosh Driver Compatibility and PrGeneral kits or from our PostScript Language SDK. For more information on these kits refer to our literature request form or call the Adobe Developers Association.

Don't Forget to register!

If you have purchased a Software Development Kit (SDK), be sure to return the registration card. We will be incorporating new features into the SDK's in the future, and want to be sure to inform you about new additions.

Macintosh Application Printing

Know what kind of printing you want your application to do

Once you understand the guidelines set down by Apple and Adobe, you should decide how your application will print. In particular, you should decide if your application will:

- generate QuickDraw calls exclusively, allowing the printer driver to do all the conversion from QuickDraw to PostScript language code,
- generate a mixture of QuickDraw and the PostScript language: the mixture may range from a small amount of PostScript language code to almost all PostScript language code,
- generate PostScript language code exclusively, bypassing the driver's conversion of QuickDraw to PostScript language output.

The amount of PostScript language functionality you would like to achieve and the amount of time you are willing to devote to developing your printing module will determine the decision concerning what combination of QuickDraw calls and PostScript language code your application should generate.

1) Exclusively generating QuickDraw

This is the simplest, least time consuming route to take. You simply take all the QuickDraw calls made during the imaging of your pages and put them inside the Macintosh print loop. The printer driver will take care of all the conversion from QuickDraw to the PostScript language. Refer to Apple's technical note PR 10 (old technical note #161): "A Printing Loop That Cares" for code samples showing how to construct a print loop. This route requires no PostScript language expertise from the application developer, since the driver is solely responsible for the PostScript language output generated.

The main caveat to exclusively generating QuickDraw from your application is that QuickDraw has many limitations. However, these limitations typically are removed if the PostScript language is used instead. The PostScript language has richer graphics capabilities and produces document descriptions that are high level, device independent and portable. The imaging and color models are relatively simple and values are described in floating-point instead of integer. To further understand the benefits of the PostScript language over QuickDraw see the first chapter of the *PostScript Language Reference Manual, Second Edition*.

Exclusively generating QuickDraw allows you to print to non-PostScript and PostScript printers. Nonetheless, you should consider generating PostScript language code when you know that the output device is indeed a PostScript printer.

2) Generating some PostScript language code: little to moderate

When you directly generate your own high-quality PostScript language code you eliminate the driver conversion time and avoid QuickDraw limitations. As a result, you will have consistently faster and better quality results when printing to PostScript printers.

continued on page 10

Colophon

This newsletter was produced entirely with Adobe PostScript software on Macintosh and IBM® PC compatible computers. Artwork was produced using Adobe Illustrator 5.0 and QuarkXPress® software. Typefaces used are from the Minion™ and Myriad™ families, all of which are from the Adobe™ Type Library.

Managing Editors:
Jennifer Cohan, Nicole Frees, Debi Hamrick

Technical Editor:
Jim DeLaHunt

Art Director:
Karla Wong

Designer:
Lorsen Koo

Contributors:
John Ciccarelli, Nicole Frees, Gayle Tyson Westbrook, Jess Walker

Part Number ADA0047 7/94

Adobe PostScript

Macintosh Application Printing

Your application may want to do very specific functions in the PostScript language. Apple recommends that you do this via the picture comments mechanism (the PicComment procedure). The recommended picture comments used in PostScript printing are PostScriptBegin, PostScriptEnd, PostScriptBeginNoSave and PostScriptHandle. These picture comments are all supported by Laserwriter versions 7.0 and later and by PSPrinter versions 8.0 and later. For details on picture comments, refer to *Inside Macintosh*, Volume I and Apple technical note QD 10 (old technical note #91): "Picture Comments—The Real Deal". You may choose to generate just a little or a lot of PostScript language code inside your print loop using picture comments.

You should avoid the PostScriptBeginNoSave picture comment. This comment instructs the driver not to put **save** and **restore** around the PostScript language code fragment your application is emitting. Its use can cause incorrect clipping, incorrect colors and possibly PostScript language errors if your code changes a graphics state that the driver depends on. If you find that you need to use the PostScriptBeginNoSave picture comment, you should consider generating all PostScript language code instead. The PostScriptFile and TextIsPostScript picture comments are obsolete. Although these comments are supported by PSPrinter 8.x, they may not be supported in future versions of the driver. For more on the recommended use of picture comments with PSPrinter, refer to Adobe Technical Note #5134: "Tips for Working with the New PostScript Printer Driver for Macintosh".

You may also want all your PostScript language procedure sets inserted into the driver output stream before the start of page printing. The LaserWriter 7.x driver and PSPrinter version 8.x insert the contents of the PREC 103 code resource into the output stream. You must place your procedure sets into this resource before your call to PrOpenPage( ) within your print loop. The driver will copy the PostScript language data from PREC 103 and put it after its procedure sets and font reencoding.

PSPrinter parses PostScript printer description files (PPDs) for a predefined set of keywords. As an application developer, you may want to provide additional printer features and options to the user. You should then consider parsing PPD files for yourself. You can find out which PPD is currently associated with the selected printer by calling PrGeneral( ) with the PSPrimaryPPD selector. See Adobe Technical Note #5134. In the PPDs, features that can be invoked with PostScript language code usually have the code right there in the file, so your application just has to put that code in the output stream using picture comments. For more information on PPD files, refer to Adobe Technical Note #5003, "PostScript Printer Description File Format Specification, Version 4.2" and the article "Supporting Printer Features" published in the April 1992 issue of *ADA News*.

When printing to a Level 2 PostScript output device, your application could generate Level 2 PostScript language code. To find out the level of the device, call PrGeneral( ) with the getPSInfo selector. See Adobe Technical Note #5134. Note that you should also generate Level 1 PostScript language code where feasible, so that the combined output stream executes on any PostScript interpreter.

Macintosh Application Printing

3) Exclusively generating PostScript language code

If you decide to take full advantage of the PostScript language and would like your application to generate all the PostScript language code sent to the printer, use the PrGeneral( ) interface provided in PSPrinter version 8.1 and later. An application that calls this interface with the kLoadCommProcsOp and kUnLoadCommProcsOp selectors can communicate directly with the printer and have complete control over all the PostScript language code and the document structuring convention (DSC) comments in the stream.

One caveat to using this interface is that the application must do *all* device set up, PPD parsing, DSC comment generation and font handling. The application will not be using the standard print loop so it will also have to control the generation of the Page Setup and Print dialogs. For a full description of how to use this PrGeneral( ) interface, refer to the Adobe Technical Note #5139: "PSPrinter's PrGeneral Interface to Get the Communications Resource".

Where to go from here

You can take the following steps to act on the information in this article:

1) Get Apple's Developer Mailing (with CD) or the Essentials•Tools•Objects Development Tools CD from APDA — (800) 282-2732 (U.S.) and (716) 871-6555 (international). These contain all the Apple documentation you will need, as well as sample code and tips on programming the Macintosh. To get Macintosh developer support, contact Apple's Developer Support Center by calling (408) 974-4897.

2) Join the Adobe Developers Association (ADA).

3) Get the Adobe Driver for Macintosh platform compatibility kit from the ADA. This kit contains the latest driver and technical notes #5133 and #5134.

4) Get the PrGeneral kit from the ADA. This kit contains sample code, the driver and technical note #5139.

5) Get the PostScript Language SDK from the ADA. This kit contains sample code, PostScript language books and many technical notes, including #5133, #5134 and #5139.

This information should get you started on understanding what your application must do to accomplish efficient printing. If you follow the guidelines documented in the technical notes, decide how your application will print best and develop high-quality PostScript language code, you'll find that your application performs better when printing to all printers, especially PostScript printers. §

Questions Answers

Q I know I can print Acrobat™ PDF files interactively from an Acrobat viewer. Is there an easy way to automate PDF printing, using a command-line utility?

A The UNIX version of Acrobat 1.0 viewers (Exchange and Reader) support command line printing if you are running an X Window System™ and have a valid license. The UNIX viewers display no graphical user interface when driven from the command line, making them convenient for use in shell scripts. The following information is from the UNIX Acrobat Exchange help file, *Help.pdf.*
Note: These examples use the `acroexch` *command, but* `acroread` *could be used instead:*

To print the file *sample.pdf* to the default printer (set by the user's LPDEST or PRINTER environment variable), enter the following command at a UNIX prompt. The print job will use Level 1 PostScript language and will print a single copy.

```
acroexch -print sample.pdf
```

Use the -P option to print to a specific PostScript printer. Note that there is no space between the P and the printer name:

```
acroexch  -print sample.pdf  -Plaserprinter
```

UNIX Acrobat viewers also work as "filters", supporting input and output redirection. In the following examples, the input redirection symbol (<) copies the file specified after the symbol into the "standard input" *(stdin)* of the command before the symbol. The output redirection symbol (>) routes the "standard output" *(stdout)* of the command before it into the file specified after it. The "pipe" symbol (|) routes the standard output of the command before it into the standard input of the command after it. These two command lines have the same effect as the first two:

```
acroexch -print < sample.pdf

acroexch -print -Plaserprinter < sample.pdf
```

If an application emits PDF on *stdout* you can pipe its output directly into a viewer to be printed.

```
pdf_generating_app | acroexch -print

pdf_generating_app | acroexch -print -Plaserprinter
```

Q What if I want to create PostScript language files from PDF files but not necessarily print them?

A You can convert PDF files to PostScript language files from the UNIX version of Acrobat Exchange 1.0 or UNIX version of Acrobat Reader 1.0 command line if you are running X Windows and have a valid license. The following command, also described in *Help.pdf*, converts several PDF files into PostScript language files and places the PostScript language files into the directory *ps_dir:*

```
acroexch -toPostScript sample1.pdf sample2.pdf sample3.pdf  ps_dir
```

To specify a PostScript language filename for each PDF filename, use the "pairs" option:

```
acroexch -toPostScript -pairs sample1.pdf sample1.ps sample2.pdf sample2.ps ...
```

To convert a PDF file on standard input to a PostScript language file on standard output, use the filter form of the *acroexch* command. The first example shows input redirection from a file; the second shows input piped from another program:

```
acroexch -toPostScript < sample.pdf > sample.ps

pdf_generating_app | acroexch -toPostScript > sample.ps
```

Note: UNIX version of Acrobat 1.0 viewers mistakenly send warnings to standard output instead of the customary standard error. If output redirection is used, any warnings will be prepended to the output file, probably making it unprintable by PostScript printers. The printer may print nothing, issue an

Questions & Answers

error report, or print the input as plain text instead of interpreting it as PostScript language code. This problem will be corrected in a future release.

You can avoid this by not using the filter form of the acroexch *or* acroread *command. If you still wish to (perhaps because this form allows input redirection), you should delete any output text before the line which begins as follows:*

```
%!PS-Adobe-3.0
```

Q Is there command-line access to the Acrobat Distiller™ application?

A On UNIX, the Distiller application is available in two forms: a command-line-only version called *distill*, and a "daemon" called *distilld* which watches directories like the Network Distiller on Mac® and Windows. Neither the UNIX version of Distiller has a graphical user interface. The following information is from *GettingStarted.pdf*, which ships with the UNIX version of Distiller.

You can distill one or more PostScript language input files and optionally place them in a different output directory as follows:

```
distill [options] [params] sample1.ps sample2.ps sample3.ps ... [pdf_dir]
```

Alternatively, the "-pairs" option lets you specify a PDF output filename for each PostScript language input filename:

```
distill [options] [params] -pairs sample1.ps sample1.pdf ...
```

Q Command-line printing and PostScript conversion are two unique features of the UNIX version of Acrobat Exchange. Are there others?

A Yes. The UNIX version of Acrobat Exchange is "email-aware". It can display PDF files included as attachments to or embedded in UNIX mail messages or mailboxes. These PDF files and the mail messages that contain them can originate from UNIX or non-UNIX mail systems.

To display a PDF file located in a mail file or mailbox, choose the File/Open command. In the Selection field enter the name of the file that holds your mail, such as `/usr/spool/mail/your-login`. Then press Return or click OK.

If Acrobat Exchange recognizes only one PDF enclosure in your mail file, that PDF file is displayed in a document window. If more than one PDF file is present, you are presented with a selection list. This information is from the UNIX version of the Exchange help file, *Help.pdf*.

Q How can I create a PDF file from a UNIX *troff* file?

A *troff* is a "text formatter" language from the days before word processors, and has historically been used to produce most UNIX documentation. The Adobe TranScript™ product suite converts the following file formats to the PostScript language, which can then be translated to PDF by the Distiller application:

troff

ditroff ("device-independent" *troff*)

ASCII text

plot

Diablo 630 print files

Tektronix 4014

To convert a *troff* file to PDF, use TranScript's *ptroff* (PostScript language from *troff*) command and pipe the resulting PostScript language output into the UNIX version of Distiller:

```
ptroff -t sample.trf | distill > sample.pdf
```

TranScript also allows you to specify fonts, page sizes, headings, and multiple columns. The above information is from *GettingStarted.pdf*, which ships with the UNIX version of Distiller. For more information on TranScript, contact Qualix at 1-800-245-8649.