New Technical Notes
Macintosh
®
Developer Support

Surprises in LaserWriter 5.0 and Newer

Imaging M.IM.LWDriverSurprise

Revised by: Mary Burke & Scott “Zz” Zimmerman February 1990
Written by: Scott “Zz” Zimmerman April 1988

This Technical Note describes some changes in version 5.0 and later LaserWriter drivers.
Changes since April 1988: Described a bug in 5.x which is fixed in 6.0 and later, and reiterated a warning about storing fonts in an application.

With the release of LaserWriter 5.0 and background printing, a few changes had to be made to the LaserWriter driver. Although these changes were transparent to most applications, some have had problems. Most of these problems are related to use of unsupported features. This Note details a partial list of the changes.

No Mo’ Low

Because of the problems supporting both the high-level and low-level interfaces in the background, the low-level interface is all but removed. Instead of the low-level calls being executed by the device driver, the `_PrCtlCall` procedure converts the call into its high-level equivalent before execution. This way, the LaserWriter driver has a common entry point for both the low-level and high-level interfaces. Because of this conversion, the low-level calls may not be faster than using the high-level equivalents. In some cases, they may even be slower.

Version 5.x of the LaserWriter driver also contains a bug with the low-level interface. If an application which uses the low-level Printing Manager interface encounters an error during the course of the print job, the LaserWriter driver crashes before the application has a chance to see the error. Because the error occurs inside the driver, there is no way for an application to predict or work around this problem. The only solution to this problem is to use the high-level Printing Manager interface or to upgrade to version 6.0 or later of the LaserWriter driver which fixes this bug.

Are You Convertible?

Whereas the conversion of the low-level calls should be transparent, the conversion routines make some assumptions. The conversion routines require a context in which to operate; the Printing Manager maintains a certain state while executing commands, and the conversion routines need access to this state to perform the conversion. To provide this context, an application must have opened a document and a page. This requirement means that the original method of using the low-level interface, which is documented in *Inside Macintosh*, Volume II-164, no longer works, as in the following example:

```
PrDrvrOpen;
PrCtlCall(iPrDevCtl, lPrReset, 0, 0);
{ Send data to be printed. }
PrCtlCall(iPrDevCtl, lPrPageEnd, 0, 0);
PrDrvrClose;
```

Instead, an application should use the following:

```
PrDrvrOpen;
PrCtlCall(iPrDevCtl, lPrDocOpen, 0, 0);
PrCtlCall(iPrDevCtl, lPrPageOpen, 0, 0);
{ Send data to be printed. }
PrCtlCall(iPrDevCtl, lPrPageClose, 0, 0);
PrCtlCall(iPrDevCtl, lPrDocClose, 0, 0);
PrDrvrClose;
```

This method provides the Printing Manager with the context it needs to convert the calls.

Really Unsupported Features

Sending data to the printer between the `_PrOpenDoc` or `lPrDocOpen` and the `_PrOpenPage` or `lPrPageOpen` calls is not currently, and has never been supported. LaserWriter drivers prior to 5.0 interpreted this data, but 5.0 and later drivers ignore it. To download an application-specific PostScript® dictionary as a header with each document, Apple recommends that the application provide a `'PREC'` resource of `ID = 103`, as is described in the *LaserWriter Reference*.

A Little Less Control

Four of the six printer control calls originally supported by the LaserWriter driver have been discontinued due to lack of use and difficulty supporting with background printing. The four calls which follow were only supported by the LaserWriter driver and only documented in the *LaserWriter Reference Manual*:

```
°•° fill     °•° hexBuf    °•° printR    °•° printF
```

In addition to these calls, the `stdBuf` call is also affected. There are two versions of the `stdBuf` call depending upon the sign of the `bytes` parameter. If `bytes` is negative, the text passed to the `stdBuf` call is converted to PostScript text before being sent to the LaserWriter. This conversion means that special PostScript characters in the text are preceded by a PostScript escape character. In addition, characters with an ASCII value greater than 128 are converted to octal before being sent to the LaserWriter. This version of the call is no longer supported.

If the bytes parameter is positive, the text passed to the call is sent directly to the LaserWriter without conversion and interpreted as PostScript instructions. This version of the call is still supported, but there is one more problem. When an application first opens the low-level driver (via `_PrDrvrOpen`) with background printing enabled, no clip region is defined. If the application then begins sending PostScript to the driver via the `stdBuf` call, all of the output is clipped, and only a blank page is printed.

To prevent this problem, the application must force a clip region to be sent to the LaserWriter. The region is sent by the driver when it receives its first drawing command. Unfortunately, the driver does not consider the `stdBuf` call to be a drawing command. To force the clip region on the printer, the application can use the `iPrBitsCtl` call to print a small bitmap outside the printable area of the page. This call does not have any effect on the document, but it fires the bottleneck routine and causes a definition of the clip region. Since the clip region is reinitialized at each call to `lPrPageOpen`, the application should send the bitmap once at the start of each page. If any other printer control calls precede the `stdBuf` call, the application does not need to send the bitmap.

Background Preparations

The `'PREC' ID = 201` mechanism only works when background printing is disabled. This limitation is because of difficulties finding the resource under MultiFinder. Since the option only works in the foreground, and since there is no way for an application to know if background printing is enabled, an application should avoid using this feature.

Fonts In An Application?

There are two problems when printing application fonts with the LaserWriter driver. *Application fonts* are fonts that are stored in the resource fork of the application's resource file rather than being stored in the System file. The first problem occurs when the application font has the same name as the application's resource file. If this is the true, the LaserWriter driver incorrectly assumes that the application's resource file is a font file, and closes it after using the font. To solve this problem, developers should make sure the name of the application font (i.e., the `'FOND'` resource) is different from the name of the application's resource file. Since the application can still be renamed by the user, developers should try to select a unique font name. If the font name does not appear in a font menu, developers can simply append the application's creator string onto the desired font name (i.e., '`MyApplicationFont ZZAP'`).

The second problem with application fonts only occurs when background printing is enabled. When a print job is performed in the background, the LaserWriter driver writes all of the data to be printed into a file (called a spool file) for printing at a later time by Print Monitor. Since the LaserWriter driver has no way of knowing when the file will actually be printed, it cannot assume that the application will still be open when the job is printed. This is a problem if the application contains application fonts that Print Monitor needs at print time. To solve this problem, the LaserWriter driver must determine whether the fonts needed by the application are resident in the System file or in the application file. If they are in the application file, the driver must copy them into the spool file so they are available to Print Monitor. This practice can lead to very large spool files, as well as a significant loss of performance when background printing is enabled. To solve these and other user interface problems related to application fonts, Apple strongly recommends that developers ship custom application fonts as suitcase files for the user to install in the System file.

Headin' For Trouble

There is a minor bug in version 5.0 of the LaserWriter driver that only affects applications that parse the PostScript header downloaded by the driver with each document. This header contains some PostScript comments that provide information about the current job. One of these comments is `IncludeProcSet`. This comment takes three arguments: a PostScript dictionary name, a major version number, and a minor version number. In version 4.0 of the LaserWriter driver, the comment line looked like the following:

```
%% IncludeProcSet: (Appledict md) 65 0
```

Unfortunately, in version 5.0 of the LaserWriter driver, the last argument was removed. This caused the comment line to look like the following:

```
%% IncludeProcSet (Appledict md) 66
```

Since Adobe defined the comment to take three arguments, it is reasonable for applications that parse the comments to expect three arguments; therefore, version 5.1 and later of the LaserWriter driver contain the correct version of the comment:

```
%% IncludeProcSet (Appledict md) 67 0
```

No Go With Zero

Some applications want to force a font to be downloaded to the LaserWriter without actually printing characters with the font. This can be done in three easy steps:

1. Save the current pen position.
2. Use any text drawing routine to draw a space character.
3. Move the pen back to the saved position.

Some applications use `_DrawString` with a empty string (e.g., `DrawString('')`) to force the font downloading. Although this worked in LaserWriter drivers up to 5.0, these calls are ignored by the 5.1 and later drivers. The main reasons for this change were optimization of performance and a reduction in the size of spool files.

Further Reference:

- *Inside Macintosh*, Volumes II & V, The Printing Manager
- *LaserWriter Reference Manual*

PostScript is a registered trademark of Adobe Systems Incorporated.