Handling Printing Errors How to handle high-level print manager errors

The Print Manager call **PrError** should be used to check for error conditions while printing. Errors returned by **PrError** will include any Print Manager errors (and some AppleTalk and OS errors) that occur during printing. Here are some error handling guidelines.

- Avoid calling **PrError** from within a PrIdle procedure. Errors that occur while PrIdle is executing are usually temporary and serve only as internal flags for communication within the printer driver. They are not intended for the application.

- If an error has been detected after the completion of a printing routine, stop where you are, that is, stop drawing. Proceed to the next print procedure to close any open calls you have made. For example, if you called **PrOpenDoc** and received an error, skip to the next **PrCloseDoc**. Remember that if a **PrOpen** call was made, you must call the corresponding **PrClose** procedure. That way, you ensure that printing closes properly and that all temporary memory allocations are released and returned to the heap.

- Do not raise any alerts or dialogs to report an error until the end of the print loop. At the end of the print loop, check for the error again; if there is no error assume that printing completed normally. If it's still there, you can raise the alert.

That last point is important for two reasons. First, if an alert is raised in the middle of the print loop, it can cause errors that will terminate an otherwise normal job. For example, if the printer is an AppleTalk printer, the connection can be terminated abnormally. While your alert is sitting there waiting for a response from the user, the driver is unable to respond to AppleTalk requests coming from the printer. If the printer doesn't hear from the Macintosh within a relatively short time (30 seconds), it will timeout. The printer assumes that the Macintosh is no longer there.

The second reason not to raise alerts within the print loop is that the driver may already have put up its own alert in response to the error. For example, if the driver detects that the version of Laser Prep that was downloaded to the LaserWriter is different from the version that the user is trying to print with, the LaserWriter driver raises the appropriate alert to tell the user and to give the option of reinitializing. If the user chooses to cancel, the driver posts an error to let the application know that it needs to abort. However, since the driver has already taken care of the error by putting up an alert, the error is reset to zero before the printing loop is complete. The application should check for the error again at the end of the printing loop and if it still indicates an error, it should raise an alert.

Version 5.x of the LaserWriter driver contains a bug with the low-level interface. If an application using the low-level Print 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 the application to predict or work around the problem. The only solution is to use the high-level Print Manager Interface or to upgrade to version 6.0 or later of the LaserWriter driver (which fixes the bug).