Sending EPS

(Encapsulated PostScript)

to a Printer

Summary:

An application can perform the following steps to send EPS (Encapsulated PostScript) to a PostScript printer as part of a document:

1. Use the DEVICEDATA printer escape to send the information.

2. Frame the EPS with the gsave and grestore commands. If the EPS contains any complicated output, use the save and restore commands instead.

The PostScript printer driver preserves the order of its input. Also, because the PostScript driver does not perform any banding, the print spooler should not cause any problems.

Note: The application must download code to the printer to correctly position the EPS image on the page [that is, to change the current transform matrix (CTM)]. In addition, to avoid unwanted page ejections, the application must edit the EPS to remove commands such as ShowPage.

Supporting PostScript Features in Windows

Summary:

There are some issues involved when designing an application to provide support for PostScript printers. The application must determine if the PostScript driver is available by using an accurate detection system. If an application generates

PostScript directly, the PASSTHROUGH escape can be used to send the file. This must be done with care because the application is communicating directly with the printer.

More Information:
The first issue is how to determine if a PostScript driver is an installed printer driver under Windows. An application cannot assume the PostScript driver is named PSCRIPT.DRV because this forces PostScript driver vendors to use the same filename.

The correct method is to run code similar to the pseudocode below:

```
    bFound = FALSE;
    for (each device in [Devices] section of win.ini) {
/* extract the necessary fields from the ini line */ szDriverName = driver name
extracted from ini line szModelName = left side of ini line (the key)
            szPort = port name extracted from ini line.
hIC = CreateIC(szDriverName, szModelName, szPort, NULL);
            if (hIC) {
                        /* see if driver supports GETTECHNOLOGY escape */
                        wEscape = GETTECHNOLOGY;
if (Escape(hIC, QUERYESCSUPPORT, sizeof(WORD), &wEscape, NULL))
{ Escape(hIC, GETTECHNOLOGY, 0, NULL, &szTechnology);
                                    /* Check that the string starts with PostScript
                                            * by doing a case-insensitive search.
Allow
                                            * for the possibility that the string could
be
* longer, like "PostScript level 2" or some other * extension.
                                            */
                                    if (beginning of string is "PostScript")
                                                bFound = TRUE;
                        }
                        DeleteDC(hIC);
                }
                /* if the driver has been found break out */
                if (bFound)
                        break;
    }
    if (bFound) {
PostScript driver is szDriverName, model is szModelName, port is szPort.
    }
```

The second issue is how to print application-generated PostScript code. The mechanism from a Windows application is through the PASSTHROUGH escape. The PASSTHROUGH escape is documented in the "Microsoft Windows Software Development Kit Reference Volume 2," Chapter 12. In addition to the

documentation, one requirement on the buffer passed is easy to miss; the first word must contain the length of the buffer. The contents of the data sent by PASSTHROUGH can alter the state of the printer.

To be safe, obey the following rules:

1. Surround PASSTHROUGH data by save/restore PostScript operators.

2. Do not embed GDI calls between PASSTHROUGH escapes. For example:

   PASSTHROUGH(save)
   Rectangle
   OtherGDIRoutines
   PASSTHROUGH(restore)

   Some driver code and software fonts are downloaded to the printer under certain conditions. The above operations could cause the driver and printer to loose synchronization, and potentially cause the job to fail. In general, no assumptions should be made concerning the code generated by a given GDI call.

3. Do not send a command to cause a page ejection.

Additional reference words: 3.00 3.10 3.x
KBCategory:
KBSubcategory: GdiPrnMisc

TrueType Support

The Windows 3.1 PostScript driver supports TrueType fonts allowing users to create and print document containing TrueType fonts on PostScript printers. The driver also provides a variety of advanced options the user determine how TrueType fonts are to be used with the printer. Depending on the capabilities of t printer, the user may want to convert TrueType fonts to an Abode format, or substitute the TrueType font printer-resident font.

The PostScript driver may need to convert the fonts to PostScript device-font format before downloading to the printer. This is true if the printer does not support the TrueType font format itself. If the driver must convert TrueType fonts, it can convert them to either Adobe Type 3 bitmap fonts or Type 1 outline fonts. The user sets the type in the TrueType option in the Advanced Options dialog box. The TrueType to Type 1 conversion is not exact and may produce fonts of slightly inferior quality to the original TrueType font. However, using this conversion will reduce print time and require much less printer memory for documents containing lots of TrueType fonts at large point sizes.
The user can direct the PostScript driver to substitute PostScript printer-resident fonts for TrueType fonts by using the Substitution option in the Advanced

Options dialog box. This option displays another dialog box that lists the TrueType fonts and lets the user choose the substitution font. Using substitution fonts reduces print time and requires
less printer memory.

If the printer supports the TrueType font format, the user can also direct the driver to download TrueType fonts to the printer by using the Substitution option in the Advanced Options dialog box. In this case, the user sets the Download as a SoftFont option for a TrueType font instead of choosing a printer-resident font. The driver converts the
Truetype font to a soft font and downloads it to the printer. This is the default for any TrueType font that hasn't been assigned a printer font in the substitution table.

End.