PostScript Printer Description Files

&

Driver Limitations

The PostScript Printer Description file format, defined by Adobe and extended by the Microsoft, provides the basis for external printer descriptions. This ASCII file contains one or more statements, each consisting of a keyword and associated values. Each keyword defines a feature of the PostScript printer. The associated values specify whether the feature is available, or how the feature affects the operation of the printer. For example, the "*FileSystem" keyword specifies whether a printer supports fonts loaded on a hard drive. PPD files must be standard ASCII files with carriage return an
d line feed pairs terminating each line.

To ensure the best performance for a printer, the PPD file should be as complete as possible. Windows 3.1 has defined new PPD keywords that should be added to existing PPD files to create new WPD files. If these keywords are not added, the driver can still use the old WPD files, but it will assign default values to any keywords not explicitly
defined.

The following provides descriptions of the PPD keywords.

Keyword	**Description**
ColorDevice	Indicates whether the printer supports color.
DefaultFont	The name of the default font (that is, the font used if none is selected). This setting must appear before any "*Font" settings.
DefaultInputSlot	The default input slot.
DefaultResolution	The default resolution of the printer.

Duplex PostScript commands to set duplex mode.
This feature is new for Windows 3.1.

FileSystem	Reserved; do not use.
Font	The fonts resident in the printer.

FreeVM Amount of free memory in standard printer configuration.
This feature is new for Windows 3.1.

ImageableArea	The actual area that can be marked on for every paper size.
InputSlot	The PostScript code that is necessary to specify each input slot.
ManualFeed False	The PostScript code that is necessary to turn off manual feed. If present, it is assumed that manual feed is supported.
ManualFeed True	The PostScript code that is necessary to specify the manual feed operation. If present, it is assumed that manual feed is supported (and therefore PageRegion settings must be included).
NickName	The name that appears in the printer dialog box. It should be a unique description of the printer. This is also the name used for automatic printer recognition.
PageRegion	The PostScript code that is necessary to specify different page sizes when using manual feed.
PageSize	The PostScript code used to specify the different page sizes that are supported (when not in manual-feed mode).
PaperDimension	The size of all the used paper types. A Paper Dimension setting must be included for every size of paper supported. The sizes of the standard page types should be used. Only standard page types are recognized. *SetResolution Used to determine the hardware resolution(s) supported. This feature is new for Windows 3.1.
Transfer Normalized	The normalized transfer function used to generate linear gray levels. This must be present. If none is required, include the nul function “{ }.”

The following is a list of PPD extensions:

Extension	Description
AcceptsTrueType:	True or false. If true, TrueType fonts will be downloaded natively using readhexsfnt operator. This is selected through the download option in the Advanced Options dialog box. If false, TrueType will not be displayed as a selection. This feature is new for Windows 3.1.
EndOfFile	Indicates whether ^D is required to indicate the end of the file. This is true by default, and only needs to be included if this is false (that is, "*EndOfFile False").
SetPage	True or false. If true, the PostScript driver allows a custom paper size to be defined. This is implemented through the use of the setpage operator. If your printer supports the setpage operator in the same manner as Linotronic® printers, you can use this option.
TrueImageDevice:	True or false. If true, the PostScript driver takes advantage of some optimizations available in TrueImage™. Currently there is very little difference in the output. This feature is new for Windows 3.1.

The following are the paper keywords used to show the paper sizes supported:

Keyword	**Description**
10x14	10 x 14 inches physical size, oriented in portrait mode.
11x17	11 x 17 inches physical size, oriented in portrait mode. Can be used interchangeably with the keyword "Tabloid."
A3	297 x 420 millimeters physical size, oriented in portrait mode. Refers to the International Standards Organization (ISO)/(JIS) A3 paper size.
A4	210 x 297 millimeters physical size, oriented in portrait mode.

A4	Extra 9.27 x 12.69 inches physical size.
A4	Small 210 x 297 millimeters physical size, but with a reduced-size imageable area of 7.47 x 10.85 inches that is centered on an A4 page. Supports the Adobe PostScript paper definitions.
A5	148 x 210 millimeters physical size, oriented in portrait mode.
B4	250 x 354 millimeters physical size, oriented in portrait mode. Refers to the Japanese Industrial Standard (JIS) B4 paper size.
B5	182 x 257 millimeters physical size, oriented in portrait mode.
Folio	8.5 x 13 inches physical size, but with a reduced-size imageable region, oriented in portrait mode and centered on the folio sheet. Supports the Adobe PostScript paper definitions.
Ledger	17 x 11 inches physical size, oriented in landscape mode (that is, the y-axis is on the shorter edge of the paper).
Legal	8.5 x 14 inches physical size, oriented in portrait mode.
LegalExtra	9.5 x 15 inches physical size.
Letter	8.5 x 11 inches physical size. Refers to the standard paper type.
LetterExtra	9.5 x 12 inches physical size.
LetterSmall	8.5 x 11 inches physical size, but with a reduced-size imageable region that is centered on the page. Supports the Adobe PostScript paper definitions.
Note	8.5 x 11 inches physical size, but with a reduced-size imageable region. This is used to reduce the size of the page buffer to give print jobs more memory.

Quarto	215 x 275 millimeters physical size, but with a reduced-size imageable region, oriented in portrait mode and centered on the quarto sheet.
Statement	5.5 x 8.5 inches physical size, oriented in portrait mode.
Tabloid	11 x 17 inches physical size, oriented in portrait or tabloid mode (that is, the y-axis is on the longer edge of the paper).
TabloidExtra	11.69 x 18 inches physical size.

For paper extensions, five standard envelope sizes are recognized. The two groups of numbers following the word Envelope indicate the size of the envelope in points (where each point equals 1/72 of an inch).

Extension	**Description**
Envelope.279.639	#9 Envelope (3.875 x 8.875 inches)
Envelope.297.684	#10 Envelope (4.125 x 9.5 inches)
Envelope.324.747	#11 Envelope (4.5 x 10.375 inches)
Envelope.342.792	#12 Envelope (4.75 x 11 inches)
Envelope.360.828	#14 Envelope (5 x 11.5 inches)

The following are the paper tray and bin keywords used to show and specify the input slots supported.

Keyword Description	
LargeCapacity	This one can hold more than a standard amount of paper.
Lower	If there is more than one tray, this one is on the bottom. Middle This one is in the middle.
OnlyOne	There is only one tray.
Upper	If there is more than one tray, this one is on top. The following list describes the paper tray extensions.
Extension	Description
AutoSelect	Printer can select automatically which feeder to use. This is followed by the code (or a nul

	command if no code is required) that is used to specify the autofeed mechanism.
Envelope	There is an envelope feeder.
EnvelopeManual	There is a manual envelope feeder.
None	There are no input feeders. This is treated as being the same as OnlyOne. The following keywords are required to support duplex printing.

Keyword	**Definition**
Duplex DuplexNoTumble:	“statusdict begin false settumble true setduplexmode end”
Duplex DuplexTumble:	“statusdict begin true settumble true setduplexmode end”
Duplex None:	“statusdict begin false setduplexmode end”
DefaultDuplex	None

If the PPD with correct *Duplex settings is built with the Windows 3.1 MKPRN utility, the driver expands the Options dialog box to give user control of duplex settings.

PostScript Fonts Have Two Font Names

A PostScript font has two font names:

1. The Windows name for the font, such as "AvantGarde," which appears in the font list in the Fonts dialog box in Control Panel.
2. The PostScript name for the font, such as "itc avant garde gothic," which can vary by printer manufacturer and which the printer driver sends to the printer to select the font.

More Information:
If the font is a downloadable soft font defined in the printer font metrics (PFM) file format, the Windows name and the PostScript name for a font can be determined. The PFM file header has a field (dfFace) that points to the Windows font name and a field (dfDriverInfo) that points to the driver-specific PostScript font name.

If the font is an internal printer font defined for a specific printer, it is necessary to enumerate fonts in a given printer driver to find the Windows font name. There is

no device-independent way to retrieve the PostScript font name for an internally defined printer font.

For more information concerning the PostScript PFM file format, refer to Chapter 4 of the "Microsoft Windows Device Development Kit Printers.

Supporting PostScript Features in Windows

The information in this article applies to:

Microsoft Windows Software Development Kit for Windows versions 3.1 and3.0

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.

Driver Limitations

The following are limitations placed on the driver by PostScript:
PostScript does not support most raster operations (Rops). However, it does support BLACKNESS, WHITENESS, and SRCCOPY.

PostScript has a 750-point polygon limit. This number is reduced by two when filling with a hatch or pattern. This is because a clipping path must be built as well as the path to fill and stroke. In cases where this limit is reached, the driver will request that GDI simulate the polygon. This is very slow. Applications should avoid generating large polygons.

End.