Font Embedding

This technical note explains in some detail the issues surrounding the embedding of fonts when using PDFDevice.

To ‘Embed’ or not to ‘Embed’?

If cross platform appearance is of vital importance to you, then you must ensure that PDFDevice embeds all fonts during printing. It is also advisable to embed fonts if you use unusual unicode characters that cannot be encoded in standard Mac Roman or Windows ANSI character sets.

If, on the other hand, the file size is of greater importance than the above criteria, perhaps font embedding is not for you and it can be turned off.

You can control overall font embedding by setting the device parameter kDevPdfEmbedFonts. This setting defaults to kTrue.

Note: Embedding font data increases the size of a PDF file, but we do recommend embedding of fonts in order to produce sound portable documents. In all unicode versions of Omnis, all true-type fonts are always embedded.

What Fonts can be Embedded?

PDFDevice can only embed true-type fonts. In today’s systems this is usually no longer a problem as most fonts are now true-type. Even most third-party fonts are now available as true-type fonts.

However, there are situations when font embedding may fail even for true-type fonts.

License Problems

Typically, most fonts installed as part of the operating system are fully licensed for sub-setting and embedding in PDF files (see below for an explanation of sub-setting). However, some fonts that have been purchased or downloaded may not permit this. A flag within the font file header information tells PDFDevice if the font allows it. By default, PDFDevice will not embed a font that prohibits the sub-setting or embedding of its data. There is however a PDFDevice parameter that you can set to ignore this state and force PDFDevice to embed such fonts. You should only override this setting if you are certain that you own a license that allows you to embed the font(s) in question. Please see ‘Font Errors and Warnings’ below, to learn how you can find out which fonts failed to be embedded.

You can control overall font embedding of license restricted fonts by setting the device parameter kDevPdfEmbedLicFonts to kTrue. This setting defaults to kFalse.

Type 1 Font Substitution

PDFDevice Version 3 implements the option to substitute non-true-type fonts with true-type fonts that have similar characteristics of the font being substituted. This feature is enabled by assigning kTrue to the device parameter kDevPdfSubstituteFonts.

WARNING: This will only work with type 1 fonts that implement standard mac-roman or Windows ANSI character sets. It may not produce the desired output for type 1 bar code or other symbolic fonts for which alternative true-type fonts should be used instead.

Style Matching Issues Macintosh

Only applies to versions prior to version 3. See "Style Matching Issues Resolved" below.

Because of the way Omnis deals with fonts and styles there is the potential of font embedding to fail due to style mismatching.

For many years, fonts have been supplied as packages of font families that contain one or more fonts for the styles supported by that family. For example, the font family 'American Typewriter' supports the following styles: Light, Regular, Bold, Condensed , Condensed Light, and Condensed Bold. Typically modern software would only offer these choices when selecting a style for 'American Typewriter' so the possible font and style combinations that you can choose are.

American Typewriter.Light

American Typewriter.Regular

American Typewriter.Bold

American Typewriter.Condensed

American Typewriter.Condensed Light

American Typewriter.Condensed Bold

Omnis however, for historical reasons, still provides separate bold and italic options for all fonts within families, whether they support them or not. Mac OS X and Windows supports what we shall call font synthesizing. That means if a font in a family does not support bold or italic, the system creates glyph outlines on the fly.

Unfortunately for PDFDevice, we cannot easily access the system's synthesized glyph data, and PDFDevice is not capable of synthesizing glyph data itself.

To make matters worse, since Omnis Studio started using the ATSUI font system on the Macintosh, Omnis font names and style options have become even more obscure. When Omnis reads information for the 'American Typewriter' font, it creates the following set of font names that are usable by the Omnis library.

American Typewriter

American Typewriter.Light

American Typewriter.Condensed

American Typewriter.Condensed Light

American Typewriter.Condensed Bold

The font 'American Typewriter.Regular' has been mapped by Omnis to 'American Typewriter' with the 'kPlain' style and the font 'American Typewriter.Bold' has been mapped to the 'American Typewriter' with the Omnis 'kBold' style.

Unfortunately all of the fonts, as they are listed above, will support the additional Omnis 'bold' and 'italic' styles. As you can see 'Italic' is simply not supported by 'American Typewriter', and neither is 'Light Bold', but if you set it, Omnis will apply it and the system will give it's version of that font in Italic or Bold or both.

It is this situation that will cause PDFDevice to fail to embed the font as PDFDevice cannot access the synthesized glyph data.

The following table shows the full set of Omnis font and style combinations and which combinations are supported by PDFDevice for the purpose of embedding.

Omnis Font Name	Omnis Style	Supported
American Typewriter	Plain - Bold - Italic	Yes - Yes - No
American Typewriter.Light	Plain - Bold - Italic	Yes - No - No
American Typewriter.Condensed	Plain - Bold - Italic	Yes - Yes - No
American Typewriter.Condensed Light	Plain - Bold - Italic	Yes - No - No
American Typewriter.Condensed Bold	Plain - Bold - Italic	Yes - No - No

The font and style combinations that are not supported by the font will, by default, not be embedded by PDFDevice. However, there is a PDFDevice option that allows one to override this behavior.

You can tell PDFDevice to ignore styles that are not supported and pick the nearest possible font for embedding. You do this by setting the device parameter kDevPdfIgnoreFontStyle to kTrue (default is kFalse). Fonts embedded in this way may not have the same appearance.

Please see 'Font Errors and Warnings' below, to learn how you can find out which fonts failed to be embedded.

Style Matching Issues Windows

Only applies to versions prior to version 3. See "Style Matching Issues Resolved" below.

For some reason (probably historical) Omnis on Windows deals with fonts and font names differently. You will not find font names that include a dot followed by the styles that are not Regular, Bold or Italic. If we look at Arial for example: For the 'Arial' font family the system lists the styles Regular, Black, Italic, Bold and Bold Italic. This gives rise to the following Font options.

Arial Regular

Arial Black

Arial Italic

Arial Bold

Arial Bold Italic

Omnis simply lists two Arial fonts; 'Arial' and 'Arial Black'. But in combination with these two fonts, Omnis offers the usual kBold and kItalic options. So theoretically, this gives rise to additional font options.

Omnis Font Name	Omnis Style	Supported
Arial	Plain - Bold - Italic	Yes - Yes - Yes
Arial Black	Plain - Bold - Italic	Yes - No - No

The style options that will fail to embed correctly are Arial Black with any combination of kBold or kItalic. Unfortunately, because of the lack of the separating dot between font family name and font style name, it is more difficult to identify font and style combinations that may fail. As a result the only option is to monitor font errors via the new feature described below.

A curious observation is that when you set the kBold option for Arial Black there is no visual difference between 'Arial Black' and 'Arial Black Bold'. It appears the system or Omnis ignores the bold setting in this case. However setting the kItalic option will render Arial Black in italic.

As with Macintosh, PDFDevice on windows will fail to embed the font when there are style mismatches unless you set the kDevPdfIgnoreFontStyle option to kTrue.

Style Matching Issues Resolved

With the release of version 3, PDFDevice now resolves style matching issues by simulating bold and italic styles when they are used with fonts that do not naturally support these styles. In such cases bold and italic are simulated using appropriate matrix settings that increase the font's glyphs fatness or slant the glyphs or both and in that way synthesizes the fonts appearance.

What is sub-setting?

Sub-setting is the art of extracting a subset of the glyph data from the font data. When embedding fonts in a PDF file it is not necessary to include all the data contained within the font. If the content of the PDF document only uses the characters A, B and C of that font, than PDFDevice will only embed the data for these three characters. This dramatically reduces the overall size of the PDF file. However, some licensed fonts may specifically prohibit the sub-setting of their font data. When licensing a font for embedding in PDF documents, please also ensure that sub-setting is permitted.

Font Errors and Warnings

In version 2.6.0 of PDFDevice we have introduced a new feature that allows one to monitor any errors and warnings associated with font embedding. The function $getfonterrors returns a list of errors and warnings in parameter one (calling this function clears the errors). The function is available from the Omnis 'Catalogue' under the functions tab, in the 'PDF Device' group.

Syntax:

```
Do PDF Device.$getfonterrors(errorList)
```

The list returned by $getfonterrors contains three columns. The error code, the error text and the postscript font name. The following list shows the possible errors

12: Font embedding error. Font is not a true type font.

13: Font embedding error. Font is not licensed for sub-setting or embedding.

14: OBSOLETE in version 3. Font style setting is not supported with this font.

15: Font embedding warning. Font licensing ignored.

16: Font embedding warning. Font style setting is not supported with this font and was synthesized.

17: Font substitution warning. Font has been substituted.

18: Font character warning. Font cannot be embedded and mapping to 8bit OS character failed.

Errors 12 and 13 indicate that the specified font has not been embedded.

Warnings 15 and 16 are reported instead of errors 13 and 14 if the device parameters kDevPdfEmbedLicFonts and/or kDevPdfIgnoreFontStyle have been enabled. The warning indicates that the font has been embedded despite the problem. For warning 16 the text appearance may not be identical to the appearance of the text in Omnis.

Warning 17 is reported if a non-true-type font was used and has been substituted by a true-type font. For this to occur, kDevPdfSubstituteFonts has to be enabled.

Warning 18 is reported when a font cannot be embedded but non standard characters (i.e. unicode characters) have been used that cannot be encoded in Win ANSII or Mac Roman character sets. Typically, these characters will be replaced by spaces to maintain the integrity of the PDF.

If at all possible try to avoid these errors and warnings by only using true-type fonts with styles that are supported by the original font family. See Summary below.

Summary

To ensure that fonts are always embedded follow these simple rules:

1. Only use true-type fonts.
2. Only use true type fonts that are licensed for sub-setting and embedding and turn on the option kDevPdfEmbedLicFonts if you have obtained permission for restricted fonts.
3. Avoid style mismatches by not allowing the use of kBold and kItalic for fonts that contain a special style in their name. On the Macintosh, such font names contain a dot followed by the name of the special style(s), i.e. ‘American Typewriter.Condensed’. On MS Windows there is no easy way of telling other than looking for known style names within a font name.
   If preventing invalid style options is not possible, you can enable the option kDevPdfIgnoreFontStyle, to force PDFDevice to use the closest embed-able font. The appearance however, may be altered. Alternatively you can monitor the font errors, see step 4.
4. Monitor all font errors and warnings (especially during development) by implementing a call to $getfonterrors after printing to PDF. You could allow your users to resolve any font issues, by changing the font options.

Document History

24 September 2013:	updated section “Font Errors and Warnings” for version 3
14 August 2013:	style matching issues resolved and type 1 font substitution
23 September 2011:	corrections
2 July 2008:	first publication