Q: My fonts don't work with the AppKit! What can I do?

A: Yikes!! This is not an easy question to answer, as fonts are very complicated and any number of things could be wrong. However, there are many things you can do to narrow the scope of the problem.

There are a couple of ways in which a malformed or misinstalled PostScript font manifests itself. Sometimes the font name simply doesn't appear in the Font panel of an application. Sometimes the new font name does appear, but when selected the display reads:

```
<<Unusable Font>>
```

And sometimes when a font is selected and previewed within the Font panel, the name of the font appears in the display, but it looks exactly like your default font (as a matter of fact, it is displayed in your default font). In this case, if you look in the console, you may likely see error messages which look like this:

```
Feb 15 12:36:57 me Edit[13537]: DPS client library error: PostScript program error, DPSContext 48160
Feb 15 12:36:58 me Edit[13537]: %%[ Error: typecheck; OffendingCommand: setshared ]%%
Feb 15 12:36:58 me Edit[13537]: DPS client library error: PostScript program error, DPSContext 48160
Feb 15 12:36:58 me Edit[13537]: %%[ Error: typecheck; OffendingCommand: setfont ]%%
```

So, how do you fix it?

First, there is a bug in the AppKit which may cause the erroneous `<<unusable font>>` message. So the first thing that you should do when debugging a newly installed font is to make sure that you are not encountering

this bug.

The following steps will cause the bug:

- Add a font to **~/Library/Fonts** (or **/LocalLibrary/Fonts**).
- In a shell, run **buildafmdir** on that directory.

There are a couple of other dependencies, but in general the new font will be `<<unusable>>`. Rebooting your system does not fix thisÐthe directory is in a bad state. The user level work-around is:

- Move a font out of the directory. Then move it back in (thus updating the modification time of the directory).
- Remove the **.afmcache**, **.fontdirectory**, and **.fontlist** files from the font directory.
- Launch an application with a Font Panel. Open the Font Panel. An alert should be displayed indicating "Incorporating info about new fonts...". All correct fonts in the directory should now be usable.

The underlying cause here is that in Releases 2 and 3 there are two programs that are run on the font directory: **buildafmdir** (which creates **.fontdirectory** and **.fontlist**) and **cacheAFMData** (which creates **.afmcache**). When the AppKit notices that there are new fonts, it does the right thing and runs both. If you just run **buildafmdir** by hand, you sometimes confuse the caching of the AFM data. The only reasons you might run **buildafmdir** yourself would be: (1) to debug fonts, or (2) because the AppKit didn't run it for

you, for some reason. This is how you run the **buildafmdir** utility by hand:

```
% buildafmdir ~/Library/Fonts
```

The alert indicating "Incorporating info about new fonts..." may not be displayed immediately after launching an application. This is because the AppKit discovers new fonts lazily. For applications that use the FontManager when they launch, new fonts are incorporated at launch. For other applications, it doesn't happen until you open the Font Panel. Opening the Font Panel always does this.

The best way to install a font is to just copy it into a font directory. In fact, it may also be best to remove **.fontdirectory**, **.fontlist**, and **.afmcache** after installing a new font, and launching a new application.

Once you have verified that the Workspace Manager has properly recognized the newly installed font, you should determine whether or not the problems that you are having lie with the installation of the font. A fontÐlet's call it **Scrabble**Ðshould have the following file system structure: there should be *at least* these three directories in **~/Library/Fonts**, or **/NeXTLibrary/Fonts** or **/LocalLibrary/Fonts** :

- **Scrabble.font**
- **afm**
- **outline**

(If any font contains bitmap fonts, then a **bitmap** directory is required as well.) The **Scrabble.font** directory

must contain all of the files which describe the font: **Scrabble and Scrabble.afm**. Any Release 2 or 3 software uses the PostScript outline, .afm, and .bepf files out of the containing .font directory, whereas the 1.0 software requires symbolic links in the outline, afm, and bitmap directories.

The file **Scrabble** contains the PostScript program which describes the outline of each individual character. The file **Scrabble.afm** contains the width information for each character. Both the outline file and the afm file contain the name of the font, and the font family among other information about the font. The file names *must* match the font names which appear within the files.

Taking our Scrabble example, in the Scrabble file we should see the following entries:

```
/FullName (Scrabble) readonly def
/FamilyName (Scrabble) readonly def
/FontName /Scrabble def
```

And in the Scrabble.afm file, we should see these entries:

```
FontName Scrabble
FullName Scrabble
FamilyName Scrabble
Weight Medium
```

The names change a bit when you consider that many font families provide both bold and italic versions.

Here's how a bold version of the Scrabble font would look:

The directory would be named **Scrabble-Bold.font**.

The outline file would be named **Scrabble-Bold**. It would contain the following entries:

```
/FullName (Scrabble Bold) readonly def
/FamilyName (Scrabble) readonly def
/FontName /Scrabble-Bold def
```

The afm file would be named **Scrabble-Bold.afm**. It would contain the following entries:

```
FontName Scrabble-Bold
FullName Scrabble Bold
FamilyName Scrabble
Weight Bold
```

It is important that the FullName be the FamilyName followed by whatever you want to appear in the second (Typeface) browser of the FontPanel. In the above example, the word "Scrabble" appears in the Family browser in the FontPanel and the word "Bold" appears in the Typeface browser. If the FullName of the font is exactly equal to the FamilyName, then the word that appears in the second column is the value of the Weight field in the afm file. (This would have been the case for the example "FullName Scrabble" above.)

And finally, on 1.0 you must link these files into the afm and outline directories. Use the following commands to do that:

```
% cd ~/Library/Fonts/afm
% ln -s ../Scrabble.font/Scrabble.afm
% cd ../outline
% ln -s ../Scrabble.font/Scrabble
```

If your font contains bitmap files, you must link those files into the bitmap directory.

Once you think that you have your fonts in order, you have to relaunch an application (and open the Font Panel) to take advantage of the new fonts. When an application is launched, the AppKit can detect that new fonts have been added to the directory, and it runs **buildafmdir** and **cacheAFMdata**, as described above.

Once you have correctly installed the font, there may still be some problems. Now, you must determine whether the problem lies within the PostScript outline file or with the afm file. The manner in which the font fails can sometime indicate which is the guilty party. If you are receiving an error in the Font panel says the font is unusable, then the afm file is guilty.

Another way to determine whether the fault is with the outline file or with the afm file is to run the outline file through **pft**.

Here's how to do that (the commands that you type are in bold):

```
myhost> pft
Connection to PostScript established.
(/Net/foobar/home/squiggle/Library/Fonts/Scrabble.font/Scrabble) run
%%[ Error: undefined; OffendingCommand:  ]%%
quit
pft: Connection to Window Server closed
myhost>
```

You may have noticed that the full pathname of the font's outline file is required. The command **pft** merely opens a connection to the WindowServer, which has been running since you booted your machine. The WindowServer is always launched from the root directory, and you must give the pathname from the current directory of the WindowServer, not your current directory.

In the above **pft** session the outline file had a syntax error in it, and the PostScript interpreter could not process it, producing an error. Of course, since PostScript error handling is archaic and does not inform you of the problem and where it exists, you must take further steps to discover just where that problem is. If you have it, you can use the DisplayTalk application to trace through the PostScript code, and it identifies the line on which the error occurs. Otherwise, you must resort to brute force method of debugging such as printing statements, and removing parts of the code a little bit at a time.

While you are still connected to **pft** you can look at the values of the font dictionary to see if things are as they should be. Refer to the Red Book (the PostScript Language Reference Manual) for more information on

the structure of the font dictionary. Remember also that fonts are stored in the FontDirectory in the PostScript interpreter.

Some common problems to look for when debugging your fonts:

· Check for strange characters like <ctrl-Z> or <ctrl-D> characters at the end of the outline file. Often in the DOS/Windows or Macintosh worlds these characters are used to indicate end-of-job to the printer. However, it's not part of the PostScript language; it's really a printer protocol, and doesn't belong in a PostScript program.

· Also, make sure that the carriage returns in the files are indeed carriage returns and not line feeds. Again, this is caused from the difference between file formats for DOS/Windows or Macintosh and those for UNIX.

· Since these files are ASCII and partially human-readable, you can look at the files. Sometimes just by looking at the files carefully, you may notice something that doesn't look right, and which may indeed be the problem. Check to make sure that entries are consistently capitalized. For example, if you saw the following:

```
FullName Scrabble
FamilyName Scrabble
weight standard
Notice
ItalicAngle 0.0
```

There's a big clue that the weight entry is incorrect.

· The Weight entry has a discrete set of valid values. They are "Ultra Light", "Thin", "Light", "Extra Light", "Book", "Regular", "Plain", "Roman", "Medium", "Demi", "Demi-Bold", "Semi-Bold", "Bold", "Extra Bold", "Heavy", "Heavyface", "Black", "Ultra", "UltraBlack", "Fat", "ExtraBlack", and "Obese".

For more information refer to the following documentation:

> *Adobe Type 1 Font Format Book*
> */NextLibrary/Documentation/NextDev/Concepts/Fonts.rtf (on-line in Digital Librarian)*
> *The Red Book (Level 1 and Level 2)*

or look at the fonts in /NextLibrary/Fonts for example fonts.

QA746

Valid for 1.0, 2.0, 3.0, 3.1