Fonts (Developer Information)

1. Goals

- refactor existing font logic for better clarity and to reduce duplication
- parse registered font metric information on-the-fly (to make sure most up-to-date parsing is used??)
- resolve whether the FontBBox, StemV, and ItalicAngle font metric information is important or not -- if so, parse the .pfb file to extract it when building the FOP xml metric file
- handle fonts registered at the operating system (through AWT)
- add support for parsing OpenType fonts

2. Issues

- Why are we using our own font metric parsing and registration system, instead of the AWT system provided as part of Java?
  - Answer 1: Many of our customers use FOP in a so-called "headless" server environment -- that is, the operating system is operating in character mode, with no concept of a graphical environment. We need some mechanism of allowing these environments to get font information. Java 1.4 has a mechanism for dealing with headless environments, and this issue may be resolved when we require that as a minimum platform. However, there may be an issue then of how to get fonts registered at the operating system in these environments. That will probably at least require some documentation for users.
  - Answer 2: At some level, we don't yet fully trust AWT to handle fonts correctly. There are still unresolved discrepancies between the two systems.
  - Answer 3: In the AWT mechanism, there does not appear to be a way to find the physical font file associated with an AWT font, or to otherwise get access to its contents so that it can be embedded in FOP output.
  - Answer 4: The Java 1.4 javadocs state (in java.awt.Font): "All implementations of the Java 2 platform must support TrueType fonts; support for other font technologies is implementation dependent." We wish to provide a greater base of font technologies for our users.

3. Implementation

There are two main font functions needed within FOP:

- provision of a font object to be used in layout
- embedding of a font file in output (such as PDF)

For the first of these, we will implement something along the lines of a "Facade" Structural Pattern to hide the differences between the various font types and font sources from the rest of the system. Public classes will consist of TypeFaceFamily, TypeFace, and Font. (TypeFace roughly corresponds to the contents of a normal font file, while Font is a general typeface implemented at a specific point size, and perhaps with other specific parameters). When another part of FOP requests a font object, existing font objects will be checked first, and an appropriate one returned if possible. If not, the Font logic should resolve the TypeFace and TypeFaceFamily if possible, create a Font object, and return it.

4. Resources

4.1. Type 1 Fonts

- Adobe Type 1 Font Format
- According to the Adobe web site, the documentation for the font metrics files (.pfm = printer font metrics) is written and controlled by Microsoft, since it is actually a workaround to allow Type 1 fonts to be used on a GUI screen in Windows. However, the document does not appear to be on the Microsoft web site. The best resource for this information is Adobe Technical Note #5178: Building PFM Files for Postscript-Language CJK Fonts
- FOP does not currently use the Adobe Font Metrics file, but the specification can be found in Adobe Technical Note #5004: Adobe Font Metrics File Format Specification
- Adobe Technical Note #5040: Supporting Downloadable Postscript Language Fonts may also include some useful information.

4.2. TrueType Fonts

- The TrueType specification

4.3. OpenType Fonts

- The OpenType specification
- The Adobe Introduction to OpenType fonts page has some useful general information and links.