Apache™ FOP: Fonts

Version 1353186

by Jeremias Märki, Tore Engvig, Adrian Cumiskey, Max Berger

Table of contents

1 Summary

The following table summarizes the font capabilities of the various Apache# FOP renderers:

Renderer	Base-14	AWT/OS	Custom	Custom Embedding
PDF	yes	no	yes	yes
PostScript	yes	no	yes	yes
PCL	yes (modified)	yes (painted as bitmaps)	yes (painted as bitmaps)	no
AFP	no	no	yes	yes
Java2D/AWT/Bitmap	if available from OS	yes	yes	n/a (display only)
Print	if available from OS	yes	yes	controlled by OS printer driver
RTF	n/a (font metrics not needed)	n/a	n/a	n/a
TXT	yes (used for layout but not for output)	no	yes (used for layout but not for output)	no
XML	yes	no	yes	n/a

2 Base-14 Fonts

The Adobe PostScript and PDF Specification specify a set of 14 fonts that must be available to every PostScript interpreter and PDF reader: Helvetica (normal, bold, italic, bold italic), Times (normal, bold, italic, bold italic), Courier (normal, bold, italic, bold italic), Symbol and ZapfDingbats.

The following font family names are hard-coded into FOP for the Base-14 font set:

Base-14 font	font families
Helvetica	Helvetica, sans-serif, SansSerif
Times	Times, Times Roman, Times-Roman, serif, any
Courier	Courier, monospace, Monospaced
Symbol	Symbol
ZapfDingbats	ZapfDingbats

Please note that recent versions of Adobe Acrobat Reader replace "Helvetica" with "Arial" and "Times" with "Times New Roman" internally. GhostScript replaces "Helvetica" with "Nimbus Sans L" and "Times" with "Nimbus Roman No9 L". Other document viewers may do similar font substitutions. If you need to make sure that there are no such substitutions, you need to specify an explicit font and embed it in the target document.

3 Missing Fonts

When FOP does not have a specific font at its disposal (because it's not installed in the operating system or set up in FOP's configuration), the font is replaced with "any". "any" is internally mapped to the Base-14 font "Times" (see above).

4 Missing Glyphs

Every font contains a particular set of glyphs. If no glyph can be found for a given character, FOP will issue a warning and use the glpyh for "#" (if available) instead. Before it does that, it consults a (currently hard-coded) registry of glyph substitution groups (see Glyphs.java in Apache XML Graphics Commons). This registry can supply alternative glyphs in some cases (like using space when a no-break space is requested). But there's no guarantee that the result will be as expected (for example, in the case of hyphens and similar glyphs). A better way is to use a font that has all the necessary glyphs. This glyph substitution is only a last resort.

5 Java2D/AWT/Operating System Fonts

The Java2D family of renderers (Java2D, AWT, Print, TIFF, PNG), use the Java AWT subsystem for font metric information. Through operating system registration, the AWT subsystem knows what fonts are available on the system, and the font metrics for each one.

When working with one of these output formats and you're missing a font, just install it in your operating system and they should be available for these renderers. Please note that this is not true for other output formats such as PDF or PostScript.

6 Custom Fonts

Support for custom fonts is highly output format dependent (see above table). This section shows how to add Type 1 and TrueType fonts to the PDF, PostScript and Java2D-based renderers. Other renderers (like AFP) support other font formats. Details in this case can be found on the page about output formats.

In earlier FOP versions, it was always necessary to create an XML font metrics file if you wanted to add a custom font. This unconvenient step has been removed and in addition to that, FOP supports auto-registration of fonts, i.e. FOP can find fonts installed in your operating system or can scan user-specified directories for fonts. Font registration via XML font metrics file is still supported and may still be necessary for some very special cases as fallback variant while we stabilize font auto-detection.

Basic information about fonts can be found at:

- Adobe font types
- Adobe Font Technote

7 Basic font configuration

If you want FOP to use custom fonts, you need to tell it where to find them. This is done in the configuration file and once per renderer (because each output format is a little different). In the basic form, you can either

tell FOP to find your operating system fonts or you can specify directories that it will search for support fonts. These fonts will then automatically be registered.

```
<renderers>
   <renderer mime="application/pdf">
      <fonts>
         <!-- register all the fonts found in a directory -->
         <directory>C:\MyFonts1</directory>

         <!-- register all the fonts found in a directory and all of its sub directories (use with
 care) -->
         <directory recursive="true">C:\MyFonts2</directory>

         <!-- automatically detect operating system installed fonts -->
         <auto-detect/>
      </fonts>
   </renderer>
</renderers>
```

> **Note:**
>
> Review the documentation for FOP Configuration for instructions on making the FOP configuration available to FOP when it runs. Otherwise, FOP has no way of finding your custom font information. It is currently not possible to easily configure fonts from Java code.

8 Advanced font configuration

The instructions found above should be sufficient for most users. Below are some additional instructions in case the basic font configuration doesn't lead to the desired results.

8.1 Type 1 Font Metrics

FOP includes PFMReader, which reads the PFM file that normally comes with a Type 1 font, and generates an appropriate font metrics file for it. To use it, run the class org.apache.fop.fonts.apps.PFMReader:

Windows:

```
java -cp build\fop.jar;lib\avalon-framework.jar;lib\commons-logging.jar;lib\commons-io.jar
          org.apache.fop.fonts.apps.PFMReader [options] pfm-file xml-file
```

Unix:

```
java -cp build/fop.jar:lib/avalon-framework.jar:lib/commons-logging.jar:lib/commons-io.jar
          org.apache.fop.fonts.apps.PFMReader [options] pfm-file xml-file
```

PFMReader [options]:

- **-fn <fontname>** By default, FOP uses the fontname from the .pfm file when embedding the font. Use the "-fn" option to override this name with one you have chosen. This may be useful in some

cases to ensure that applications using the output document (Acrobat Reader for example) use the embedded font instead of a local font with the same name.

Note:

The classpath in the above example has been simplified for readability. You will have to adjust the classpath to the names of the actual JAR files in the lib directory. xml-apis.jar, xercesImpl.jar, xalan.jar and serializer.jar are not necessary for JDK version 1.4 or later.

Note:

The tool will construct some values (FontBBox, StemV and ItalicAngle) based on assumptions and calculations which are only an approximation to the real values. FontBBox and Italic Angle can be found in the human-readable part of the PFB file or in the AFM file. The PFMReader tool does not yet interpret PFB or AFM files, so if you want to be correct, you may have to adjust the values in the XML file manually. The constructed values however appear to have no visible influence.

8.2 TrueType Font Metrics

FOP includes TTFReader, which reads the TTF file and generates an appropriate font metrics file for it. Use it in a similar manner to PFMReader. For example, to create such a metrics file in Windows from the TrueType font at c:\myfonts\cmr10.ttf:

```
java -cp build\fop.jar;lib\avalon-framework.jar;lib\commons-logging.jar;lib\commons-io.jar
         org.apache.fop.fonts.apps.TTFReader [options]
         C:\myfonts\cmr10.ttf ttfcm.xml
```

TTFReader [options]:

- **-d <DEBUG | INFO >** Sets the debug level (default is INFO).
- **-fn <fontname>** Same as for PFMReader.
- **-ttcname <fontname>** If you're reading data from a TrueType Collection (.ttc file) you must specify which font from the collection you will read metrics from. If you read from a .ttc file without this option, the fontnames will be listed for you.
- **-enc ansi** Creates a WinAnsi-encoded font metrics file. Without this option, a CID-keyed font metrics file is created. The table below summarizes the differences between these two encoding options as currently used within FOP. Please note that this information only applies to TrueType fonts and TrueType collections:

Issue	WinAnsi	CID-keyed
Usable Character Set	Limited to WinAnsi character set, which is roughly equivalent to iso-8889-1.	Limited only by the characters in the font itself.
Embedding the Font	Optional.	Mandatory. Not embedding the font produces invalid PDF documents.

Warning:

You may experience failures with certain TrueType fonts, especially if they don't contain the so-called Unicode "cmap" table. TTFReader can currently not deal with font like this.

8.3 TrueType Collections

TrueType collections (.ttc files) contain more than one font. To create metrics files for these fonts, you must specify which font in the collection should be generated, by using the "-ttcname" option with the TTFReader.

To get a list of the fonts in a collection, just start the TTFReader as if it were a normal TrueType file (without the -ttcname option). It will display all of the font names and exit with an Exception.

Here is an example of generating a metrics file for a .ttc file:

```
java -cp build\fop.jar;lib\avalon-framework.jar;lib\commons-logging.jar;lib\commons-io.jar
          org.apache.fop.fonts.apps.TTFReader -ttcname "MS Mincho"
          msmincho.ttc msminch.xml
```

Alternatively, the individual sub-fonts of a TrueType Collections can be selected using the "sub-font" attribute on the "font" element. That means that generating an XML font metrics file for TrueType collections is not necessary anymore. Example:

```
<font embed-url="gulim.ttc" sub-font="GulimChe">
  <font-triplet name="GulimChe" style="normal" weight="normal"/>
</font>
```

8.4 Register Fonts with FOP

You must tell FOP how to find and use the font metrics files by registering them in the FOP Configuration. Add entries for your custom fonts, regardless of font type, to the configuration file in a manner similar to the following:

```
<renderers>
   <renderer mime="application/pdf">
      <fonts>
         <!-- register a particular font -->
         <font metrics-url="file:///C:/myfonts/FTL_____.xml" kerning="yes"
           embed-url="file:///C:/myfonts/FTL_____.pfb"
           encoding-mode="single-byte">
            <font-triplet name="FrutigerLight" style="normal" weight="normal"/>
         </font>

         <!-- register all the fonts found in a directory -->
         <directory>C:\MyFonts1</directory>

         <!-- register all the fonts found in a directory and all of its sub directories (use with
 care) -->
         <directory recursive="true">C:\MyFonts2</directory>

         <!-- automatically detect operating system installed fonts -->
         <auto-detect/>
      </fonts>
   </renderer>
</renderers>
```

- URLs are used to access the font metric and font files. Relative URLs are resolved relative to the font-base property (or base) if available. See FOP: Configuration for more information.
- The "metrics-url" attribute is generally not necessary except if you run into problems with certain fonts.
- Either an "embed-url" or a "metrics-url" must be specified for font tag configurations.

- The font "kerning" attribute is optional. Default is "true".
- If embedding is off (i.e. embed-url is not set), the output will position the text correctly (from the metrics file), but it will not be displayed or printed correctly unless the viewer has the applicable font available to their local system.
- When setting the "embed-url" attribute for Type 1 fonts, be sure to specify the PFB (actual font data), not PFM (font metrics) file that you used to generate the XML font metrics file.
- The attribute "encoding-mode" is optional an may have the following values:
  - auto: default font encoding mode ("cid" for Truetype, "single-byte" for Type 1)
  - single-byte: use single-byte encodings in the target format (if applicable)
  - cid: encode as CID-keyed font (currently only supported for PDF output with TrueType fonts)
- The fonts "directory" tag can be used to register fonts contained within a single or list of directory paths. The "recursive" attribute can be specified to recursively add fonts from all sub directories.
- The fonts "auto-detect" tag can be used to automatically register fonts that are found to be installed on the native operating system.
- Fonts registered with "font" tag configurations override fonts found by means of "directory" tag definitions.
- Fonts found as a result of a "directory" tag configuration override fonts found as a result of the "auto-detect" tag being specified.
- If relative URLs are specified, they are evaluated relative to the value of the "font-base" setting. If there is no "font-base" setting, the fonts are evaluated relative to the base directory.

8.5 Auto-Detect and auto-embed feature

When the "auto-detect" flag is set in the configuration, FOP will automatically search for fonts in the default paths for your operating system.

FOP will also auto-detect fonts which are available in the classpath, if they are described as "application/x-font" in the MANIFEST.MF file. For example, if your .jar file contains font/myfont.ttf:

```
Manifest-Version: 1.0

          Name: font/myfont.ttf
          Content-Type: application/x-font
```

This feature allows you to create JAR files containing fonts. The JAR files can be added to fop by providem them in the classpath, e.g. copying them into the lib/ directory.

8.5.1 The font cache

Apache FOP maintains a cache file that is used to speed up auto-detection. This file is usually found in the ".fop" directory under the user's home directory. It's called "fop-fonts.cache". When the user's home directory is not writable, the font cache file is put in the directory for temporary files.

If there was a problem loading a particular font, it is flagged in the cache file so it is not loaded anymore. So, if a font is actually around but is still not found by Apache FOP, it's worth a try to delete the font cache file which forces Apache FOP to reparse all fonts.

8.6 Embedding

By default, all fonts are embedded if an output format supports font embedding. In some cases, however, it is preferred that some fonts are only referenced. When working with referenced fonts it is important to be in control of the target environment where the produced document is consumed, i.e. the necessary fonts have to be installed there.

There are two different ways how you can specify that a font should be referenced:

1. When using the old-style "font" element to configure a single font, font referencing is controlled by the embed-url attribute. If you don't specify the embed-url attribute the font will not be embedded, but will only be referenced.
2. For automatically configured fonts there's a different mechanism to specify which fonts should be referenced rather than embedded. This is done in the "referenced-fonts" element in the configuration. Here's an example:

```
<fop version="1.0">
   <fonts>
      <referenced-fonts>
        <match font-family="Helvetica"/>
        <match font-family="DejaVu.*"/>
      </referenced-fonts>
   </fonts>
</fop>
```

At the moment, you can only match fonts against their font-family. It is possible to use regular expressions as is shown in the second example above ("DejaVu.*"). The syntax for the regular expressions used here are the one used by the `java.util.regex` package. So, in the above snippet "Helvetica" and all variants of the "DejaVu" font family are referenced. If you want to reference all fonts, just specify `font-family=".*"`.

The `referenced-fonts` element can be placed either inside the general `fonts` element (right under the root) or in the `fonts` element under the renderer configuration. In the first case, matches apply to all renderers. In the second case, matches only apply to the renderer where the element was specified. Both cases can be used at the same time.

Various notes related to embedded fonts:

- The font is simply embedded into the output file, it is not converted.
- When FOP embeds a font in a PDF file, it adds a prefix to the fontname to ensure that the name will not match the fontname of an installed font. This is helpful with older versions of Acrobat Reader that preferred installed fonts over embedded fonts.
- When embedding PostScript fonts, the entire font is always embedded.
- When embedding TrueType fonts (ttf) or TrueType Collections (ttc), a subset of the original font, containing only the glyphs used, is embedded in the output document. That's the default, but if you specify encoding-mode="single-byte" (see above), the complete font is embedded.

8.7 Substitution

When a <substitutions/> section is defined in the configuration, FOP will re-map any font-family references found in your FO input to a given substitution font.

- If a <substitution/> is declared, it is mandatory that both a <from/> and <to/> child element is declared with a font-family attribute.
- Both font-weight and font-style are optional attributes, if they are provided then a value of 'normal' is assumed.

For example you could make all FO font-family references to 'Arial' with weights between 700 and 900 reference the normal 'Arial Black' font.

```
<fop version="1.0">
   <fonts>
      <substitutions>
         <substitution>
            <from font-family="Arial" font-weight="700..900"/>
            <to font-family="Arial Black"/>
         </substitution>
         <substitution>
            <from font-family="FrutigerLight"/>
            <to font-family="Times" font-weight="bold" font-style="italic"/>
         </substitution>
      </substitutions>
   </fonts>
</fop>
```

9 Font Selection Strategies

There are two font selection strategies: character-by-character or auto. The default is auto.

Auto selected the first font from the list which is able to display the most characters in a given word. This means (assume font A has characters for abclmn, font B for lnmxyz, fontlist is A,B):

- aaa lll xxx would be displayed in fonts A A B
- aaaxx would be displayed in font A
- aaaxxx would be displayed in font A
- aaaxxxx would be displayed in font B

Character-by-Character is NOT yet supported!

10 Font List Command-Line Tool

FOP contains a small command-line tool that lets you generate a list of all configured fonts. Its class name is: `org.apache.fop.tools.fontlist.FontListMain`. Run it with the "-?" parameter to get help for the various options.