Using non-Western fonts in your Firebird docs

A font embedding guide for Firebird docwriters and translators

Paul Vinkenoog

7 March 2020 – Document version 1.1

Table of Contents

Introduction

If you use non-Western characters in your Firebird documentation, the HTML version will probably look fine – modern browsers can display any language and script, as long as the necessary fonts are present on the user's machine. But making the PDF version come out right requires a number of extra steps. This is what you have to do:

1. Find the fonts you need for body text, titles, and monospaced text.
2. Override the default (Western) fonts specified in the stylesheets.
3. Create *metrics files* for the fonts you're going to use.
4. Create a *FOP userconfig file* with font embedding instructions.
5. Build the PDF and if everything works as intended, commit the results of steps 2–4 to git.

If you are the first person in the Firebird documentation project to write in a certain language, you'll have to complete all of the above steps. Otherwise you can use the existing setup, but you may have to edit the FOP userconfig file because font file locations can differ from system to system. See the "Important" box in the section on Step 4.

It is assumed that you already know how to create and edit DocBook XML files in your own language and save them in a Unicode encoding, e.g. in UTF-8. XMLMind, SciTE, Windows Notepad and others have no problem with this (XMLMind saves in UTF-8 by default). ConText can only save Unicode as UTF-16, but as far as I know this poses no problem for the build tools.

How the PDF is created

To better understand the effects of the steps you're going to take, here's a brief overview of how your DocBook source is converted to a PDF file.

First stage: DocBook -> XSL-FO

A so-called *XSL transformer* by the name of Saxon reads the DocBook XML source, converts it to *XSL-FO* format, and saves it in the `manual/inter/fo` tree. The FO in XSL-FO stands for *Formatting Objects*. Like DocBook, this is an XML format, but presentation-oriented. Below is a typical chunk of XSL-FO:

```
<fo:block keep-together="always" margin-left="0pc"
          font-family="sans-serif,Symbol,ZapfDingbats">
  <fo:block font-family="sans-serif" font-weight="bold"
            keep-with-next.within-column="always"
            space-before.minimum="0.8em" space-before.optimum="1.0em"
            space-before.maximum="1.2em"
            color="darkblue" text-align="start">
    <fo:block font-size="19.8pt">SQL Syntax</fo:block>
  </fo:block>
</fo:block>
```

To learn *how* we want each piece of DocBook converted to XSL-FO, Saxon also loads the transformation stylesheets. The default DocBook stylesheets are downloaded by the build, and our own customisations are in `src/docs/xsl`.

This first stage, the DocBook -> XSL-FO transformation, is executed by the `docbookFo` build target. If you give a "**gradlew docbookPdf...**" command, the `docbookFo` target is called internally, but you can also call it explicitly to generate only the XSL-FO file and not the PDF.

Second stage: XSL-FO -> PDF

The XSL-FO file is converted to PDF by another tool, Apache FOP (*Formatting Objects Processor*). The result is stored in the `build/pdf-`*`baseName`* tree. Stylesheets don't enter into this, nor does the original DocBook source.

This stage is represented by the `docbookPdf` build target.

The sections to come will guide you through the steps listed in the Introduction. This table shows where each of them affects the DocBook-to-PDF build process:

Step	Description	Affects
1	Finding the fonts you need	(preparatory step)
2	Overriding the stylesheet fonts	Stage 1: XSL-FO generation
3	Creation of metrics files	Stage 2: PDF production
4	Creation of FOP userconfig file	Stage 2: PDF production
5	PDF building and committing	(final step – the build itself)

Step 1: Find the fonts you need

The DocBook stylesheets distinguish six font "families": for body, title, monospaced, symbol, dingbat and "sans" (= sans-serif) text. They are defined as follows:

```
body.font.family      = serif
title.font.family     = sans-serif
monospace.font.family = monospace
symbol.font.family    = Symbol,ZapfDingbats
dingbat.font.family   = serif
sans.font.family      = sans-serif
```

The sans.font.family isn't used in practice; it is only listed for completeness. The symbol and dingbat families should probably not be changed. That leaves you with the body, title and monospace families to find suitable fonts for.

For English and other Western languages, "serif" translates to Times New Roman in the PDF, "sans-serif" to Helvetica/Arial, and "monospace" to Courier. These fonts, as well as the Symbol font, are supported by all Adobe PDF readers. That's why Western docwriters don't have to take any special measures to get their language represented correctly. But most non-Western characters aren't present in the Adobe standard fonts. If you don't supply the fonts yourself, the tools will build the PDF without complaints, but every character not present in the standard font will be replaced with a "#", making your text look like this: #### ## #### #### ##### ## ## #####.

It is advised that you choose fonts that are widely present on computer systems in your country. Metrics and configuration files are committed to CVS and can be reused by every doc builder, but the font files themselves are not. They must be present – and locatable – on the user's system every time a PDF is built. (Note: only when it's *built*, not when it's *read*!)

If your language or script doesn't make a distinction between serif and sans-serif fonts, feel free to pick the same font for the body and title families – or use another distinction which is appropriate in your language. But do try to use another font for the monospace family, even if the difference between proportional (variable-width) and monospaced (fixed-width) fonts is meaningless in your situation. The point is that monospaced text is often used within sentences, to make certain words stand out from the rest.

Every font chosen should support these styles and weights: normal, italic, bold, and bold italic. (Again, in as far as these are valid concepts in your language.) Sometimes these variations are housed in one single font file, sometimes they are distributed over up to four files. Only Type 1 and TrueType fonts can be used.

Because of the tremendous differences between the world's many languages and scripts, it's impossible to give any more specific instructions here. But feel free to discuss any problems or questions regarding this on the firebird-docs list.

Once you've decided on the fonts to use, make a note of the file locations. TrueType font files typically have the extension `.ttf`, TrueType collections `.ttc`. Type 1 font files have extensions `.pfb` (the font itself) and `.pfm` (the metrics info). To embed Type 1 fonts, you need both the `.pfb` and the `.pfm`.

Step 2: Override the stylesheet fonts

Any font configuration stuff is placed in the folder `config/xx`, where `xx` is your language code. If the folder doesn't exist yet, create it. Now edit the file `fo-params.txt` in this folder (if it doesn't exist, copy it from `config`). Suppose you are working on a Japanese setup and you've chosen the following fonts: MSGothic for titles, MSMincho for normal body text, and WPJapanese to replace monospaced text. You would then edit the relevant portion of `manual/config/ja/fo-params.txt` as follows:

```
body.font.family=MSMincho
title.font.family=MSGothic
monospace.font.family=WPJapanese
```

Make sure you uncomment each altered line, if necessary.

Now when the intermediate XSL-FO output is built for a Japanese document, it will contain references to these fonts instead of the default ones in the stylesheets. A piece of the `.fo` file may look like this:

```
<fo:block keep-together="always" margin-left="0pc"
          font-family="MSMincho,Symbol,ZapfDingbats">
  <fo:block font-family="MSMincho" font-weight="bold"
            keep-with-next.within-column="always"
            space-before.minimum="0.8em" space-before.optimum="1.0em"
            space-before.maximum="1.2em"
            color="darkblue" text-align="start">
    <fo:block font-size="19.8pt">...Japanese text here...</fo:block>
  </fo:block>
</fo:block>
```

The next two steps will deal with the following stage: the creation of the PDF itself.

Step 3: Create metrics files

For every non-standard font you use, a file with font metrics information must be generated. Apache FOP needs this information while performing the XSL-FO `->` PDF conversion. You can create the metrics files with the

`t1Metrics` and `ttfMetrics` targets in our build system. At the command prompt in the root of the firebird-documentation project, type (on one line!):

```
gradlew ttfMetrics
  --fontFile=D:\Path\To\fontfile.ttf --metricsFileName=filename.xml
```

to build a TTF metrics file, or

```
gradlew t1Metrics
  --fontFile=D:\Path\To\fontfile.pfm --metricsFileName=filename.xml
```

for a Type 1 metrics file.

Please note:

- You must provide the full path to the font file, but only a file name for the metrics file. The metrics file will be placed in `build/font-metrics`. You will need to manually copy it to the appropriate language directory or directories in `config`.
- You are free in your choice of the metrics file's base name, but it's wise to make it indicative of the font.
- For the `t1Metrics` target, you must specify the `.pfm` file, not the `.pfb`.
- The second character of `t1Metrics` is the digit `1` (one), not the letter `el`.
- An additional parameter *`fontName`* exists to override the font name to use when embedding. For example the MS Mincho font will default to use MS-Mincho as the embedded name, but we override it to MSMincho.

Remember that you must repeat this step for every font you add. If the bold and/or italic variations reside in different files, you must also create a separate metrics file for each variation.

> **Important**
>
> Metrics files created with older version of FOP (which was part of our build tools until March 2020) are not usable with our current FOP version (2.4). If you have such files in your tree, regenerate them with the latest tool set. Notice however that performing a git update may already bring you the current metrics files.

Font collections

Some TrueType fonts are packed together in `.ttc` files (TrueType collections). The `ttfMetrics` target allows you to build metrics files for such fonts:

```
gradlew ttfMetrics --fontFile=D:\Path\To\collection.ttc
  --ttcName=fontname --metricsFileName=filename.xml
```

There's an additional *`ttcName`* (font name) parameter. If the font name contains a space, make sure to enclose the value in double quotes. To find out which fonts are in a collection, call the `ttfMetrics` target without the *`ttcName`* parameter and with an additional *`info`* parameter, like this:

```
gradlew ttfMetrics --fontFile=D:\Path\To\collection.ttc
  --metricsFileName=filename.xml --info
```

This will result in a build failure, but just before that you'll find a list of all the fonts contained in the collection.

Step 4: Create a FOP userconfig file

This is the most complicated step. You have to edit the `fop-userconfig.xml` file to tell FOP:

- which fonts (and variations) are to be included;
- where the font files can be found;
- where the metrics files can be found.

OK, let's go for it:

1. If necessary, copy `fop-userconfig.xml` from the general `config` dir into your language subdirectory (e.g. `config/ja`).

2. Open the file in a text or XML editor and find the `font-base` entry. Replace "`xx`" in the value with your language code, so that the URL points to the correct config subdirectory. Uncomment the entry!

3. Now go on to the `<fonts>` element. You will find some commented-out example fonts already present.

4. Insert a `<font>` element for the first font to add:

```
<font metrics-url="msmincho.xml" kerning="yes"
      embed-url="file:///D:/WINNT/Fonts/MSMincho.ttf">
  <font-triplet name="MSMincho" style="normal" weight="normal"/>
</font>
```

   Notes:

   - `metrics-url` refers to the font metrics file you've created before, and which is in the same folder as `fop-userconfig.xml`.

   - `embed-url` must be a URL pointing to the font file itself. Attention! For Type 1 fonts, you must specify the `.pfb` file here, not the `.pfm` like you did when creating the metrics file.

     > **Important**
     >
     > Even if the entire setup already exists, it may be necessary to edit the `embed-url`, as it is possible that you have the font in another location than the person who committed the setup to CVS. Other than that, no configuration changes should be necessary.

   - The `font-triplet name` must be the same as the name you used in `fo-params.txt` to override the default font.

5. Now you have to add information for the bold, italic, and bold-italic variations of the font. Often these come from different font files and you will have generated separate metrics files for them. If that is indeed the case, each variation gets its own `<font>` entry, e.g. for bold-italic:

```
<font metrics-url="msmincho-bi.xml" kerning="yes"
      embed-url="file:///D:/WINNT/Fonts/MSMinchoBI.ttf">
  <font-triplet name="MSMincho" style="italic" weight="bold"/>
</font>
```

   Note that the `font-triplet name` must be the same for each variation: the name you used in `fo-params.txt`.

Sometimes there are no bold or italic variations of a font. In that case you must “fake it”, because the intermediate `.fo` file does specify these variations in places, and if they don't seem to exist you'll get the dreaded ### #### ## in the PDF again. For every non-existing variation, add a `<font-triplet>` element to the variation that should be used instead:

```
<font metrics-url="msmincho.xml" kerning="yes"
      embed-url="file:///D:/WINNT/Fonts/MSMincho.ttf">
  <font-triplet name="MSMincho" style="normal" weight="normal"/>
  <font-triplet name="MSMincho" style="italic" weight="normal"/>
</font>
```

If neither bold nor italic exist, you'll end up with four `<font-triplet>` children in the `<font>` element.

6. Repeat items 4 and 5 for every font family you've added to `fo-params.txt`.

If everything has gone right, you should now be able to generate PDFs in your language.

Important

The older FOP userconfig files (which we used until February 2020) don't have the right format for our current FOP version (2.4 or higher). If FOP complains about this, do a git update. If the format still isn't right, look in `config/fop-userconfig.xml` and give your localised userconfig file the same structure.

Step 5: Build the PDF and commit your work

To put your configuration to the test, build a PDF in your language, e.g.:

```
gradlew docbookPdf --docId=qsg15-ru --language=ru
```

Inspect the result carefully. If you find any “### #### ##” spots in the document, the location may give you a clue as to what went wrong:

- If they are in the titles, you probably forgot to override the *`title.font.family`* in `fo-params.txt` and/or to add the bold or bold-italic variations in `fop-userconfig.xml`.
- If they occur in some places in the body text, you may have forgotten to replace the *`monospace.font.family`*.
- If the document mostly consists of “#”s but the titles are OK, chances are that you didn't supply a *`body.font.family`* (this should make you feel *really* silly!)
- If the pound signs are in isolated places, maybe you have to override the Symbol and/or Dingbat families after all.

Comparing your PDF to the English original may also help to find the cause of the problem. And of course, there's always the `firebird-docs` list.

Once everything works fine, add and commit the following to git:

- The `config/xx` language subdirectory (if it's not in git yet).
- The `fo-params.txt` file in that subdirectory.
- Any `.xml` metrics files in that subdirectory.
- The `fop-userconfig.xml` file in that subdirectory.

If you don't have git write access, ask a subproject member to commit your work for you, or fork the firebird-documentation project, commit your changes on a branch and submit a pull request on GitHub.

Afterword

We are still relatively new to translating Firebird docs to non-Western scripts, and I suppose we'll have to learn a lot from experience. Please post any error reports, comments and suggestions to the `firebird-docs` list. Good luck with your docwriting and translation efforts – it's great being on this subproject!

Appendix A: Document history

The exact file history is recorded in the `manual` module in our CVS tree; see http://sourceforge.net/cvs/?group_id=9028

Revision History

0.1	22 Dec 2005	PV	First edition.
0.1.1	23 Dec 2005	PV	Small correction in Step 3 section.
0.1.2	24 Dec 2005	PV	Removed incorrect font name requirement; added notice about editing embed-file URL.
0.1.3	25 Jan 2007	PV	*How the PDF is created*: Removed a few words from the last sentence, just above the steps table.
1.0	18 Apr 2007	PV	*Create metrics files* and *Create a FOP userconfig file*: Brought up to date with FOP 0.93. *Afterword*: Changed wording. We're not complete newbies anymore :-)
1.1	7 Mar 2020	MR	Updated instructions for use with the Gradle build and update to FOP 2.4.

Appendix B: License notice

The Original Documentation is titled *Using non-Western fonts in your Firebird docs*.

The Initial Writer of the Original Documentation is: Paul Vinkenoog.

Contributor: Mark Rotteveel – see document history.