Apache FOP: Graphics Formats

$Revision: 562887 $

Table of contents

1. Overview of Graphics Support

The table below summarizes the *theoretical* support for graphical formats within FOP. In other words, within the constraints of the limitations listed here, these formats *should* work. However, many of them have not been tested, and there may be limitations that have not yet been discovered or documented. The packages needed to support some formats are not included in the FOP distribution and must be installed separately. Follow the links in the "Support Thru" column for more details.

Format	Type	FOP native support	Batik SVG	Batik codecs	Image I/O	JAI	JIMI
BMP (Microsoft Windows Bitmap)	bitmap	X					
EPS (Encapsulat PostScript)	metafile (both bitmap and vector), probably most frequently used for vector drawings	(X)					
GIF (Graphics Interchange Format)	bitmap	X			X	X	X
JPEG (Joint Photograph Experts Group)	bitmap	(X)			X		
PNG (Portable Network Graphic)	bitmap			X	X		

SVG (Scalable Vector Graphics)	vector (with embedded bitmaps)		X				
TIFF (Tag Image Format File)	bitmap	(X)		X	X	X	
EMF (Windows Enhanced Metafile)	vector (with embedded bitmaps)	(X)					

Note:
"(X)" means restricted support. Please see the details below.

2. Graphics Packages

2.1. FOP Native

FOP has native ability to handle some graphic file formats.

2.2. "Internal" codecs

Apache XML Graphics Commons contains codecs for PNG and TIFF access. FOP can use these.

2.3. Image I/O (JDK 1.4 or higher)

For JDKs 1.4 or higher, FOP provides a wrapper to load images through the JDK's Image I/O API (JSR 015). Image I/O allows to dynamically add additional image codecs. An example of such an add-on library are the JAI Image I/O Tools available from Sun.

2.4. JIMI

Because of licensing issues, the JIMI image library is not included in the FOP distribution. First, download and install it. Then, copy the file "JimiProClasses.zip" from the archive to {fop-install-dir}/lib/jimi-1.0.jar. Please note that FOP binary distributions are compiled with

JIMI support, so there is no need for you to build FOP to add the support. If jimi-1.0.jar is installed in the right place, it will automatically be used by FOP, otherwise it will not.

2.5. JAI (Java Advanced Imaging API)

FOP has been compiled with JAI support, but JAI is not included in the FOP distribution. To use it, install JAI, then copy the jai_core.jar and the jai_codec.jar files to {fop-install-dir}/lib. JAI is much faster than JIMI, but is not available for all platforms. See What platforms are supported? on the JAI FAQ page for more details.

2.6. Apache Batik

Current FOP distributions include a distribution of the Apache Batik version 1.6. It is automatically installed with FOP. Because Batik's API changes frequently, it is highly recommended that you use the version that ships with FOP, at least when running FOP.

Warning:

Batik must be run in a graphical environment.

Batik must be run in a graphical environment. It uses AWT classes for rendering SVG, which in turn require an X server on Unixish systems. If you run a server without X, or if you can't connect to the X server due to security restrictions or policies (a so-called "headless" environment), SVG rendering will fail.

Here are some workarounds:

- If you are using JDK 1.4, start it with the `-Djava.awt.headless=true` command line option.
- Install an X server which provides an in-memory framebuffer without actually using a screen device or any display hardware. One example is Xvfb.
- Install a toolkit which emulates AWT without the need for an underlying X server. One example is the PJA toolkit, which is free and comes with detailed installation instructions.

3. BMP

FOP native support for BMP images is limited to the RGB color-space.

4. EPS

FOP provides support for two output targets:

- PostScript (full support).
- PDF (partial support). Due to the lack of a built-in PostScript interpreter, FOP can only embed the EPS file into the PDF. Acrobat Reader will not currently display the EPS (it doesn't have a PostScript interpreter, either) but it will be shown correctly when you print the PDF on a PostScript-capable printer. PostScript devices (including GhostScript) will render the EPS correctly.

Warning:

Please note that the EPS embedding feature has been **deprecated** in the PDF specification version 1.4. You should not use this feature anymore, especially since newer PDF tools don't support embedded EPS files anymore.

Other output targets can't be supported at the moment because FOP lacks a PostScript interpreter. Furthermore, FOP is not able to parse the preview bitmaps sometimes contained in EPS files.

5. JPEG

FOP native support of JPEG does not include all variants, especially those containing unusual color lookup tables and color profiles. If you have trouble with a JPEG image in FOP, try opening it with an image processing program (such as Photoshop or Gimp) and then saving it. Specifying 24-bit color output may also help. For the PDF and PostScript renderers most JPEG images can be passed through without decompression. User reports indicate that grayscale, RGB, and CMYK color-spaces are all rendered properly.

6. PNG

If using JAI for PNG support, only RGB and RGBA color-spaces are supported for FOP rendering.

Transparency is supported but not guaranteed to work with every output format.

7. SVG

7.1. Introduction

FOP uses Apache Batik for SVG support. This format can be handled as an `fo:instream-foreign-object` or in a separate file referenced with

`fo:external-graphic`.

Note:

Batik's SVG Rasterizer utility may also be used to convert standalone SVG documents into PDF. For more information please see the SVG Rasterizer documentation on the Batik site.

7.2. Placing SVG Graphics into PDF

The SVG is rendered into PDF by using PDF commands to draw and fill lines and curves. This means that the graphical objects created with this remain as vector graphics. The same applies to PostScript output. For other output formats the SVG graphic will be converted to a bitmap image.

There are a number of SVG things that cannot be converted directly into PDF. Parts of the graphic such as effects, patterns and images are inserted into the PDF as a raster graphic. The resolution of these raster images can be controlled through the "target resolution" setting in the configuration.

Currently transparency is limited in PDF so many svg images that contain effects or graphics with transparent areas may not be displayed correctly.

7.3. Placing SVG Text into PDF and PostScript

If possible, Batik will use normal PDF or PostScript text when inserting text. It does this by checking if the text can be drawn normally and the font is supported. This example svg text.svg / text.pdf shows how various types and effects with text are handled. Note that tspan and outlined text are not yet implemented.

Otherwise, text is converted and drawn as a set of shapes by Batik, using the stroking text painter. This means that a typical character will have about 10 curves (each curve consists of at least 20 characters). This can make the output files large and when it is viewed the viewer may not normally draw those fine curves very well (In Adobe Acrobat, turning on "Smooth Line Art" in the preferences will fix this). If the text is inserted into the output file using the inbuilt text commands it will use a single character.

Note that because SVG text can be rendered as either text or a vector graphic, you may need to consider settings in your viewer for both. The Acrobat viewer has both "smooth line art" and "smooth text" settings that may need to be set for SVG images to be displayed nicely on your screen (see Edit / Preferences / Display). This setting will not affect the printing of your document, which should be OK in any case, but will only affect the quality of the screen display.

7.4. Scaling

Currently, SVG images are rendered with the dimensions specified *in the SVG file*, within the viewport specified in the fo:external-graphic element. For everything to work properly, the two should be equal. The SVG standard leaves this issue as an implementation detail. FOP will probably implement a scaling mechanism in the future.

If you use pixels to specify the size of an SVG graphic the "source resolution" setting in the configuration will be used to determine the size of a pixel. The use of pixels to specify sizes is discouraged as they may be interpreted differently in different environments.

7.5. Known Problems

- Soft mask transparency is combined with white so that it looks better on pdf 1.3 viewers but this causes the soft mask to be slightly lighter or darker on pdf 1.4 viewers.
- There is some problem with a gradient inside a pattern causing a PDF error when viewed in acrobat 5.
- Text is not always handled correctly, it may select the wrong font especially if characters have multiple fonts in the font list.
- More PDF text handling could be implemented. It could draw the string using the attributed character iterator to handle tspans and other simple changes of text.
- JPEG images are not inserted directly into the pdf document. This area has not been implemented yet since the appropriate method in batik is static.
- Uniform transparency for images and other svg elements that are converted into a raster graphic are not drawn properly in PDF. The image is opaque.

8. TIFF

FOP-native TIFF support is limited to PDF and PostScript output only. Also, according to user reports, FOP's native support for TIFF is limited to images with the following characteristics (all must be true for successful rendering):

- single channel images (i.e., bi-level and grayscale only)
- uncompressed images, or images using CCITT T.4, CCITT T.6, or JPEG compression
- images using white-is-zero encoding in the TIFF PhotometricInterpretation tag

Note:

Native support in this case means that the images can be embedded into the output format without decoding it.

JAI: Supports RGB and RGBA only for FOP rendering.

9. EMF

Windows Enhanced Metafiles (EMF) are only supported in RTF output.

10. Graphics Resolution

Some bitmapped image file formats store a dots-per-inch (dpi) or other resolution values. FOP tries to use this resolution information whenever possible to determine the image's intrinsic size. This size is used during the layout process when it is not superceeded by an explicit size on fo:external-graphic (content-width and content-height properties).

Please note that not all images contain resolution information. If it's not available 72 dpi is assumed (the default resolution of PDF and PostScript).

Bitmap images are generally embedded into the output format at their original resolution (as is). No resampling of the image is performed. Explicit resampling is on our wishlist, but hasn't been implemented, yet. Bitmaps included in SVG graphics may be resampled to the resolution specified in the "target resolution" setting in the configuration if SVG filters are applied. This can be used as a work-around to resample images in FO documents.

11. Image caching

FOP caches images between runs. There is one cache per FopFactory instance. The URI is used as a key to identify images which means that when a particular URI appears again, the image is taken from the cache. If you have a servlet that generates a different image each time it is called with the same URL you need to use a constantly changing dummy parameter on the URL to avoid caching.

The image cache has been improved considerably in the redesigned code. Therefore, a resetCache() method like in earlier versions of FOP has become unnecessary. If you still experience OutOfMemoryErrors, please notify us.