Apache FOP: Configuration

$Revision: 924860 $

Table of contents

1. Configuration File Basics

The FOP configuration file is an XML file containing a variety of settings that are useful for controlling FOP's behavior, and for helping it find resources that you wish it to use.

The easiest way to get started using a FOP configuration file is to copy the sample found at `{fop-dir}/conf/fop.xconf` to a location of your choice, and then to edit it according to your needs. It contains templates for the various configuration options, most of which are commented out. Remove the comments and change the settings for entries that you wish to use. Be sure to follow any instructions, including comments which specify the value range. Also, since the configuration file is XML, be sure to keep it well-formed.

1.1. Making Configuration Available to FOP

After creating your configuration file, you must tell FOP how to find it:

- If running FOP from the command-line, see the "-c" command-line option in Running FOP.
- If running FOP as an embedded application, see Embedding, Using a Configuration File.

See Setting the Configuration Programmatically for instructions on how to do so in an embedded environment.

2. Summary of the General Configuration Options

Element	Data Type (for the value)	Description	Default Value
base	URL or directory	Specifies the base URL based on which relative URL will be resolved.	current directory
font-base	URL or directory	Specifies the base URL based on which relative font URLs will be resolved.	base URL/directory (above)
hyphenation-base	URL or directory	Specifies the base URL based on which relative URLs to hyphenation pattern files will be resolved. If not specified, support for user-supplied hyphenation patterns remains disabled.	disabled
source-resolution	Integer, dpi	Resolution in dpi (dots per inch) which is used	72 dpi

		internally to determine the pixel size for SVG images and bitmap images without resolution information.	
target-resolution	Integer, dpi	Resolution in dpi (dots per inch) used to specify the output resolution for bitmap images generated by bitmap renderers (such as the TIFF renderer) and by bitmaps generated by Apache Batik for filter effects and such.	72 dpi
strict-configuration	Boolean (true, false)	Setting this option to 'true' will cause FOP to strictly verify the contents of the FOP configuration file to ensure that defined resources (such as fonts and base URLs/directories) are valid and available to FOP. Any errors found will cause FOP to immediately raise an exception.	false
strict-validation	Boolean (true, false)	Setting this option to 'false' causes FOP to be more forgiving about XSL-FO validity, for example, you're allowed to specify a border on a region-body which is supported by some FO implementations but is non-standard. Note that such a border would currently have no effect in Apache FOP.	true
break-indent-inheritance	Boolean (true, false)	Setting this option to 'true' causes FOP to use an alternative rule set to determine text indents specified through margins,	false

		start-indent and end-indent. Many commercial FO implementations have chosen to break the XSL specification in this aspect. This option tries to mimic their behaviour. Please note that Apache FOP may still not behave exactly like those implementations either because FOP has not fully matched the desired behaviour and because the behaviour among the commercial implementations varies. The default for this option (i.e. false) is to behave exactly like the specification describes.	
default-page-settings	n/a	Specifies the default width and height of a page if "auto" is specified for either or both values. Use "height" and "width" attributes on the default-page-settings element to specify the two values.	"height" 11 inches, "width" 8.26 inches
prefer-renderer	boolean (true, false)	By default, FOP prefers the newer output implementations based on the `IFDocumentHandler` interface. If no such implementation can be found for a given MIME type, it looks for an implementation of the `Renderer` interface. If necessary, you can invert the lookup order to prefer the Renderer variant over the IFDocumentHandler variant by setting this value to true.	false

use-cache	boolean (true, false)	All fonts information that has been gathered as a result of "directory" or "auto-detect" font configurations will be cached for future rendering runs. This setting should improve performance on systems where fonts have been configured using the "directory" or "auto-detect" tag mechanisms. By default this option is switched on.	true
cache-file	String	This options specifies the file/directory path of the fop cache file. This option can also be specified on the command-line using the -cache option. This file is currently only used to cache font triplet information for future reference.	${base}/conf/fop.cache
renderers	(see text below)	Contains the configuration for each renderer. See below.	N/A

This is an excerpt from the example configuration file coming with FOP:

```
<fop version="1.0">

  <!-- Strict user configuration -->
  <strict-configuration>true</strict-configuration>

  <!-- Strict FO validation -->
  <strict-validation>true</strict-validation>

  <!-- Base URL for resolving relative URLs -->
  <base>./</base>

  <!-- Font Base URL for resolving relative font URLs -->
  <font-base>./</font-base>

  <!-- Source resolution in dpi (dots/pixels per inch) for determining the size of
pixels in SVG and bitmap images, default: 72dpi -->
  <source-resolution>72</source-resolution>
  <!-- Target resolution in dpi (dots/pixels per inch) for specifying the target
resolution for generated bitmaps, default: 72dpi -->
  <target-resolution>72</target-resolution>
```

```
  <!-- default page-height and page-width, in case
       value is specified as auto -->
  <default-page-settings height="11in" width="8.26in"/>

  <!-- etc. etc..... -->
</fop>
```

3. Image Loading Customization

Apache FOP uses the image loading framework from Apache XML Graphics Commons to load images using various plug-ins. Every image loader plug-in has a hard-coded usage penalty that influences which solution is chosen if there are multiple possibilities to load an image. Sometimes, though, these penalties need to be tweaked and this can be done in the FOP configuration. An example:

```
<fop version="1.0">
  [..]
  <image-loading>
    <penalty value="10000"
      class="org.apache.xmlgraphics.image.loader.impl.ImageLoaderRawCCITTFax"/>
    <penalty value="INFINITE"
      class="org.apache.xmlgraphics.image.loader.impl.ImageLoaderInternalTIFF"/>
  </image-loading>
  <renderers....
</fop>
```

The first penalty element increases the penalty for the raw CCITT loader. This practically forces the decoding of CCITT compressed TIFF images except if there are no TIFF codecs available.

The second penalty element sets an "infinite" penalty for the TIFF loader using the internal TIFF codec. This practically disables that plug-in as it will never be chosen as a possible solution.

Negative penalties are possible to promote a plug-in but a negative penalty sum will be treated as zero penalty in most cases. For more details on the image loading framework, please consult the documentation there.

4. Renderer configuration

Each Renderer has its own configuration section which is identified by the MIME type the Renderer is written for, ex. "application/pdf" for the PDF Renderer.

The configuration for the PDF Renderer could look like this:

```
  <renderers>
    <renderer mime="application/pdf">
      <filterList>
        <!-- provides compression using zlib flate (default is on) -->
        <value>flate</value>
      </filterList>
      <fonts>
        <font metrics-url="arial.xml" kerning="yes" embed-url="arial.ttf">
          <font-triplet name="Arial" style="normal" weight="normal"/>
          <font-triplet name="ArialMT" style="normal" weight="normal"/>
        </font>
```

```
      <font metrics-url="arialb.xml" kerning="yes" embed-url="arialb.ttf">
        <font-triplet name="Arial" style="normal" weight="bold"/>
        <font-triplet name="ArialMT" style="normal" weight="bold"/>
      </font>
    </fonts>
  </renderer>

  <renderer mime="application/postscript">
  <!-- etc. etc..... -->
```

The details on the font configuration can be found on the separate Fonts page. Note especially the section entitled Register Fonts with FOP.

4.1. Special Settings for the PDF Renderer

The configuration element for the PDF renderer contains two elements. One is for the font configuration (please follow the link above) and one is for the "filter list". The filter list controls how the individual objects in a PDF file are encoded. By default, all objects get "flate" encoded (i.e. simply compressed with the same algorithm that is also used in ZIP files). Most users don't need to change that setting. For debugging purposes, it may be desired not to compress the internal objects at all so the generated PDF commands can be read. In that case, you can simply use the following filter list. The second filter list (type="image") ensures that all images still get compressed but also ASCII-85 encoded so the produced PDF file is still easily readable in a text editor.

```
  <renderer mime="application/pdf">
    <filterList>
      <value>null</value>
    </filterList>
    <filterList type="image">
      <value>flate</value>
      <value>ascii-85</value>
    </filterList>

    <fonts....
  </renderer>
```

Another (optional) setting specific to the PDF Renderer is an output color profile, an ICC color profile which indicates the target color space the PDF file is generated for. This setting is mainly used in conjunction with the PDF/X feature. An example:

```
  <renderer mime="application/pdf">
    <filterList...

    <output-profile>C:\FOP\Color\EuropeISOCoatedFOGRA27.icc</output-profile>

    <fonts....
    </renderer>
```

Some people don't have high requirements on color fidelity but instead want the smallest PDF file sizes possible. In this case it's possible to disable the default sRGB color space which XSL-FO requires. This will cause RGB colors to be generated as device-specific RGB. Please note that this option is unavailable (and will cause an error) if you enable PDF/A or PDF/X functionality or if you specify an output profile. This setting will make the PDF about 4KB smaller. To disable the sRGB color space add the following setting:

```
<renderer mime="application/pdf">
  <filterList...

  <disable-srgb-colorspace>true</disable-srgb-colorspace>

  <fonts....
  </renderer>
```

FOP supports encryption of PDF output, thanks to Patrick C. Lankswert. This feature is commonly used to prevent unauthorized viewing, printing, editing, copying text from the document and doing annotations. It is also possible to ask the user for a password in order to view the contents. Note that there already exist third party applications which can decrypt an encrypted PDF without effort and allow the aforementioned operations, therefore the degree of protection is limited. For further information about features and restrictions regarding PDF encryption, look at the documentation coming with Adobe Acrobat or the technical documentation on the Adobe web site.

```
<renderer mime="application/pdf">
  <encryption-params>
     <user-password>testuserpass</user-password>
     <owner-password>testownerpass</owner-password>
     <noprint/>
     <nocopy/>
     <noedit/>
     <noannotations/>
  </encryption-params>
</renderer>
```

4.2. Special Settings for the PostScript Renderer

Besides the normal font configuration (the same "fonts" element as for the PDF renderer) the PostScript renderer has an additional setting to force landscape pages to be rotated to fit on a page inserted into the printer in portrait mode. Set the value to "true" to activate this feature. The default is "false". Example:

```
<renderer mime="application/postscript">
  <auto-rotate-landscape>true</auto-rotate-landscape>

  <fonts>
    <font metrics-url="arial.xml" kerning="yes" embed-url="arial.ttf">
      <font-triplet name="Arial" style="normal" weight="normal"/>
      <font-triplet name="ArialMT" style="normal" weight="normal"/>
    </font>
    <font metrics-url="arialb.xml" kerning="yes" embed-url="arialb.ttf">
      <font-triplet name="Arial" style="normal" weight="bold"/>
      <font-triplet name="ArialMT" style="normal" weight="bold"/>
    </font>
  </fonts>
</renderer>
```

4.3. Special Settings for the PCL Renderer

Non-standard fonts for the PCL renderer are made available through the Java2D subsystem which means that you don't have to do any custom font configuration in this case but you have to use the font names offered by Java.

Additionally, there are certain settings that control how the renderer handles various elements.

```
<renderer mime="application/vnd.hp-PCL">
  <rendering>quality</rendering>
  <text-rendering>bitmap</text-rendering>
</renderer>
```

The default value for the "rendering" setting is "speed" which causes borders to be painted as plain rectangles. In this mode, no special borders (dotted, dashed etc.) are available. If you want support for all border modes, set the value to "quality" as indicated above. This will cause the borders to be painted as bitmaps.

The default value for the "text-rendering" setting is "auto" which paints the base fonts using PCL fonts. Non-base fonts are painted as bitmaps through Java2D. If the mix of painting methods results in unwelcome output, you can set this to "bitmap" which causes all text to be rendered as bitmaps.

4.4. Special Settings for the AFP Renderer

Additionally, there are certain settings that control how the renderer handles various elements.

```
<renderer mime="application/x-afp">
  <images mode="b+w" bits-per-pixel="8" native="true"/>
  <renderer-resolution>240</renderer-resolution>

  <!-- a default external resource group file -->
  <resource-group-file>resources.afp</resource-group-file>
</renderer>
```

The default value for the images "mode" setting is "b+w" (black and white). When the images "mode" setting is "b+w" a "bits-per-pixel" setting can be provided to aid the grayscale conversion process. With this setting all images referenced in your source document are converted to an IOCA FS45 grayscale bitmap image form. When the setting is "color" all images are converted to an IOCA FS45 color bitmap image form. When "native" setting is "true", all images encountered (TIFF, GIF, JPEG and Encapsulated Postscript etc.) will be embedded directly in the datastream in their native form using a MO:DCA Object Container.

The default value for the "renderer-resolution" is 240 dpi.

By default if there is no configuration definition for "resource-group-file", external resources will be placed in a file called resources.afp.

5. When it does not work

FOP searches the configuration file for the information it expects, at the position it expects. When that information is not present, FOP will not complain, it will just continue. When there is other information in the file, FOP will not complain, it will just ignore it. That means that when your configuration information is in the file but in a different XML element, or in a different XML path, than FOP expects, it will be silently ignored.

Check the following possibilities:

- The format of the configuration file has changed considerably between FOP 0.20.5 and FOP 1.0 and

its beta versions. Did you convert your file to the new format?

- The FOP distribution contains a schema for configuration files, at src/foschema/fop-configuration.xsd. Did you validate your configuration file against it? Add the following schema location to the `schema` element:
```
<fop
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=
"http://svn.apache.org/viewvc/xmlgraphics/fop/trunk/src/foschema/fop-configuration.xsd?
```
and run the configuration file through a validating schema parser. Note that the schema cannot detect all errors, and that it is stricter about the order of some elements than FOP itself is.
- Run FOP in debug mode (command line option `-d`). This makes FOP report which configuration information it finds. Check if FOP finds what you expect.