Intermediate Format

$Revision$

Table of contents

Note:
Please note that the intermediate format is an **advanced feature** and can be ignored by most users of Apache FOP.

1. Introduction

The intermediate format (IF) is a proprietary XML format that represents the area tree generated by the layout engine. The area tree is conceptually defined in the XSL-FO specification in chapter 1.1.2. The IF can be generated through the area tree XML Renderer (the XMLRenderer).

The intermediate format can be used to generate intermediate documents that are modified before they are finally rendered to their ultimate output format. Modifications include adjusting and changing trait values, adding or modifying area objects, inserting prefabricated pages, overlays, imposition (n-up, rotation, scaling etc.). Multiple IF files can be combined to a single output file.

2. Usage of the Intermediate Format

As already mentioned, the IF is generated by using the **XMLRenderer** (MIME type: **application/X-fop-areatree**). So, you basically set the right MIME type for the output format and process your FO files as if you would create a PDF file. However, there is an important detail to consider: The various Renderers don't all use the same font sources. To be able to create the right area tree for the ultimate output file, you need to create the IF file using the right font setup. This is achieved by telling the XMLRenderer to mimic another renderer. This is done by calling the XMLRenderer's mimicRenderer() method with an instance of the ultimate target renderer as the single parameter. This has a consequence: An IF file rendered with the Java2DRenderer may not look as expected when it was actually generated for the PDF renderer. For renderers that use the same font setup, this restriction does not apply (PDF and PS, for example). Generating the intermediate format file is the first step.

The second step is to reparse the IF file using the **AreaTreeParser** which is found in the org.apache.fop.area package. The pages retrieved from the IF file are added to an AreaTreeModel instance from where they are normally rendered using one of the available Renderer implementations. You can find examples for the IF processing in the `examples/embedding` directory in the FOP distribution

The basic pattern to parse the IF format looks like this:

```
FopFactory fopFactory = FopFactory.newInstance();
```

```
// Setup output
OutputStream out = new java.io.FileOutputStream(pdffile);
out = new java.io.BufferedOutputStream(out);
try {
    //Setup fonts and user agent
    FontInfo fontInfo = new FontInfo();
    FOUserAgent userAgent = fopFactory.newFOUserAgent();

    //Construct the AreaTreeModel that will received the individual pages
    AreaTreeModel treeModel = new RenderPagesModel(userAgent,
            MimeConstants.MIME_PDF, fontInfo, out);

    //Parse the IF file into the area tree
    AreaTreeParser parser = new AreaTreeParser();
    Source src = new StreamSource(myIFFile);
    parser.parse(src, treeModel, userAgent);

    //Signal the end of the processing. The renderer can finalize the target
document.
    treeModel.endDocument();
} finally {
    out.close();
}
```

This example simply reads an IF file and renders it to a PDF file. Please note, that in normal FOP operation you're shielded from having to instantiate the FontInfo object yourself. This is normally a task of the AreaTreeHandler which is not present in this scenario. The same applies to the AreaTreeModel instance, in this case an instance of a subclass called RenderPagesModel. RenderPagesModel is ideal in this case as it has very little overhead processing the individual pages. An important line in the example is the call to `endDocument()` on the AreaTreeModel. This lets the Renderer know that the processing is now finished.

The intermediate format can also be used from the command-line by using the "-atin" parameter for specifying the area tree XML as input file. You can also specify a "mimic renderer" by inserting a MIME type between "-at" and the output file.

2.1. Concatenating Documents

This initial example is obviously not very useful. It would be faster to create the PDF file directly. As the ExampleConcat.java example shows you can easily parse multiple IF files in a row and add the parsed pages to the same AreaTreeModel instance which essentially concatenates all the input document to one single output document.

2.2. Modifying Documents

One of the most important use cases for the intermediate format is obviously modifying the area tree XML before finally rendering it to the target format. You can easily use XSLT to process the

IF file according to your needs. Please note, that we will currently not formally describe the intermediate format. You need to have a good understanding its structure so you don't create any non-parseable files. We may add an XML Schema and more detailed documentation at a later time. You're invited to help us with that.

2.3. Advanced Use

The generation of the intermediate format as well as it parsing process has been designed to allow for maximum flexibility and optimization. Please note that you can call `setTransformerHandler()` on XMLRenderer to give the XMLRenderer your own TransformerHandler instance in case you would like to do custom serialization (to a W3C DOM, for example) and/or to directly modify the area tree using XSLT. The AreaTreeParser on the other side allows you to retrieve a ContentHandler instance where you can manually send SAX events to to start the parsing process (see `getContentHandler()`).