jPDFImposition Manual

Version 0.2 – January 2015

Maurizio M. Gavioli

Table of Contents

1. What *jPDFImposition* is

jPDFImposition is a programme to manipulate PDF files by applying an imposition and generating compound pages for printing.

It is a *console* programme, without a fancy graphical user interface (at least, not yet): all is done through command line parameters. Available parameters are documented in this manual.

Is is a Java programme and this means that it should run as it is on any computer where a Java run time is installed. Java is available on all major (and not so major…) platforms.

jPDFImposition is a free, open-source application. The latest versions of both the runnable application and the source code are available for free downloading at: https://github.com/mgavioli/jPDFImposition .

2. Legalese…

jPDFImposition relies on the **jPod** open-source library for PDF manipulation. jPod is distributed under the following license:

INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The jPod library is included in jPDFImposition distribution. Its own source code and documentation can be downloaded from: http://opensource.intarsys.de/home/en/index.php?n=JPod.HomePage.

3. Installing *jPDFImposition*

1. Download the programme from https://github.com/mgavioli/jPDFImposition/releases
2. Expand the archive (`jPDFImposition_x_x_x.zip`) in any folder you like, as long as you have full access to it; your home directory `/home/<yourUserName>` (in Linux) or the `C:\\Users\<yourUserName>` folder (in Windows) are fine. Remember to check the "Recreate sub-folders" option before expanding: *jPDFImposition* comes with some libraries in a sub-folder, all the files should always remain together in the original relationship.

   This is all: the programme is ready to run.

4. Running *jPDFImposition*

The easiest way to run *jPDFImposition* is through the supplied console scripts:

TO BE DONE

5. Command line parameters

The programme accepts the following command line parameters:

```
[options] [input-PDF-file] [output-PDF-file]
```

where [options] can be any or all of:

`-f <format name>`	One of: `booklet`, `in4h`, `in4v`, `in8h`, `in8v`, `in16h`, `in16h`
`-s nnn`	For the `booklet` format only: the number of sheets in each signature
`-l <parameter-list.xml>`	An XML file with more parameters. The parameter list can specify much more parameters than the command line. The structure of such a file is described below.

A typical command line may look like:

```
-f booklet -s 5 my-document.pdf my-document-bklt.pdf
```

6. Some details on impositions

The *imposition* is the process of (and the result of) manipulating a source document,

placing several source pages together and generating an output document, usually with larger sized pages, ready to be printed.

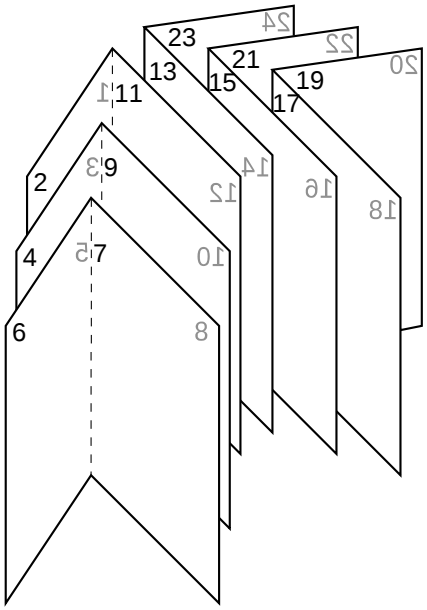
Once printed, the larger pages can be folded (and often also bound together and trimmed) to make a finished book.

Each group of pages to be folded (and bound) together is called a *signature*.

jPDFImposition knows about two main kinds of imposition:

Booklet:

Booklet pages are twice as wide as the source pages and each contains two source pages; a sheet (with its front and back pages) contains four source pages. Booklets are usually made of signatures with several sheets each, which are folded together. The following sketch shows a booklet made of two signatures of three sheets each, for a total of 24 pages:

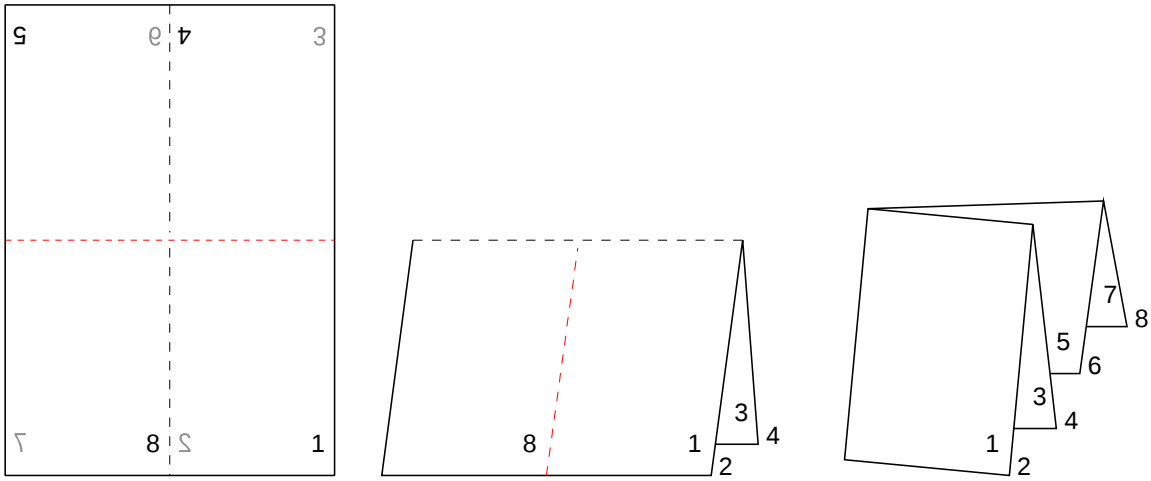


Standard impositions:

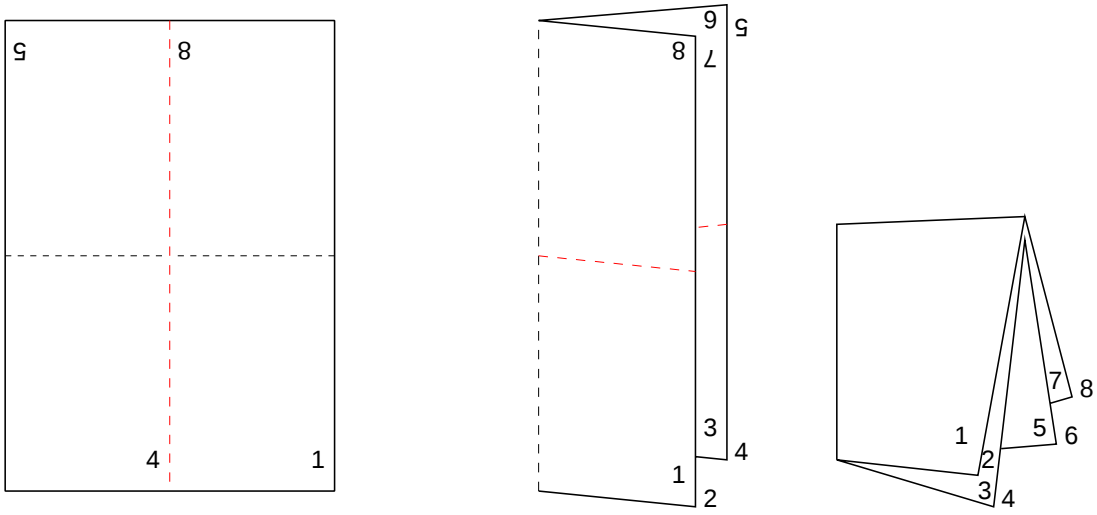
Traditionally, impositions were made out of rather large sheets, folded several times, resulting in many pages per sheet; in this kind of imposition, signatures are always made of just one sheet, with as many pages as come out of one sheet.

With two folds, each sheet side yields four pages, this is traditionally called *in 4°*, three folds yield 8 pages per sheet side (*in 8°*); four folds yield 16 pages (*in 16°*) and so on.

The first fold may occur horizontally (with respect to the orientation of the final pages) or vertically, usually depending on the paper *grain*. The following sketches show the resulting sheet of an *in 4°* imposition with first horizontal or vertical fold:



in 4° with first horizontal fold



In 4° with first vertical fold

jPDFImposition support the following standard impositions:

- `in4h`: *in 4°*, horizontal first
- `in4v`: *in 4°*, vertical first
- `in8h`: *in 8°*, horizontal first
- `in8v`: in 8°, vertical first (rather unusual)
- `in16h`: *in 16°*, horizontal first
- `in16v`: *in 16°*, vertical first

Traditionally, very large books were made out of sheets with just one fold, the so called *in folio*. *In folio* can be achieved by using the `booklet` format with 1 sheet per signature.

7. Parameter file

By giving a parameter file in the command line (option `-l parameter_file.xml`), more parameters can be specified.

7.1. Basic structure

A valid parameter file is an XML file and shall contain at least the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<jPDFImposition>
</jPDFImposition>
```

Actual parameters are inserted between the opening and closing `<jPDFImposition>` and `</jPDFImposition>` tags.

All tag names are case-insensitive. If a parameter duplicates an option and both are given, the parameter in the parameter file takes precedence.

7.2. <input> tag

Defines a source PDF file, to which the imposition will be applied. Example:

```
<input value="myfile.pdf" />
```

Several `<input>` tags can be used to concatenate several input files into a single output PDF.

If the file name value is not an absolute path (i.e. does not begin with "`/`" under *nix systems or with "*x*`://`" under Windows), it will be considered relative to the path of the parameter file being read.

Whenever a page is referenced via its page number (for instance, in `<foldout>` tags, see below), *jPDFImposition* assumes by default that page 1 is the first document page, page 2 is the second and so on. If the source document(s) have a different pagination – for instance, if they begin with page 128 or have some initial unnumbered pages – converting the page numbers into physical page indices may be cumbersome and lead to errors.

The `pageNoOffset` attribute of the `<input>` tag can be used to tell how to convert page numbers into page sequence indices. For instance, if the source document has four unnumbered pages (extra pages) at the beginning (let's say, some front matter), the following tag:

```
<input value="myfile.pdf" pageNoOffset="4" />
```

allows to use printed page numbers directly. Vice versa, if the document(s) begin with a page number greater than 1, say 25, (24 missing pages) the following tag can be used:

```
<input value="myfile.pdf" pageNoOffset="-24" />
```

7.3. <output> tag

Defines the destination PDF file, with the imposition applied. Example:

```
<output value="myfile-booklet.pdf" />
```

If the file name value is not an absolute path (i.e. does not begin with "`/`" under *nix systems or with "*x*`://`" under Windows), it will be considered relative to the path of the parameter file being read.

7.4. <format> tag

Defines the imposition format. Example:

```
<format value="in4h" />
```

The possible values are:

- `booklet`: the booklet format
- `in4h`: *in 4°*, horizontal first
- `in4v`: *in 4°*, vertical first
- `in8h`: *in 8°*, horizontal first
- `in8v`: *in 8°*, vertical first (rather unusual)
- `in16h`: *in 16°*, horizontal first
- `in16v`: *in 16°*, vertical first
- `none`: to copy the input files into the output PDF without any imposition; useful to just concatenate several PDFs as they are.

Any other value defaults to the `booklet` format.

7.5. <sheetsPerSign> tag

Only valid with the `booklet` format. Defines how many sheets each signature will contain **at most**. The actual number of sheets in a signature can be lower, because there might not be enough pages to fill the last signature and because sheets are balanced across signatures. For instance: a booklet with 4 sheets per signature is requested out of a document with 24 pages; as each booklet sheet holds 4 pages and each signature cannot have more than 4 sheets, 6 sheets and 2 signatures are needed; but, instead of creating one signature of 4 sheets and one of 2 sheets, *jPDFImposition* will balance the result into 2 signatures of 3 sheets (12 pages) each.

Example:

```
<sheetsPerSign value="5" />
```

7.6. <frontOffsetX>, <frontOffsetY>, <backOffsetX>, <backOffsetY> tags

These parameters allow to displace the front and back sides of the resulting sheet along the X and Y axis. This might be useful if the printer is not able to print the two sides in perfect register. All values are in mm. Examples:

```
<backOffsetX value="1.5" />
<backOffsetY value="2" />
```

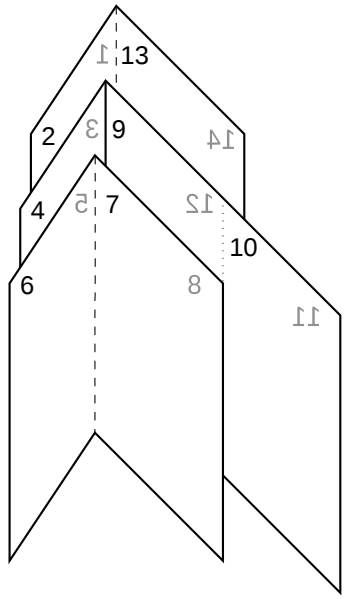
moves the back side of all the sheets by 1.5 mm to the right (positive X) and 2 mm up (positive Y).

The front side of each sheet is the one containing the lowest-numbered source page; the back side is obviously the other!

7.7. <foldout> tag

Only valid with the `booklet` format. Tells *jPDFImposition* that two pages (one leaf) of

the source document are extra pages to be folded out of the main sequence. The following sketch gives an example:



Here, pages 10 and 11 make a fold-out. Note that gluing always happens at spine position, not at edge position: pages 9 to 12 are combined into a complete sheet, while pages 3 and 4 are extracted from the sequence.

Out-of-sequence pages (pages 3 and 4 in the above example) will be appended to the end of the output document, to be printed on single-sized paper. One of each pair (page 4 in the above example) will receive a small mark telling the user to which page it is intended to be glued (in the above example, it would be “p. 9”).

Either page of the fold-out can be indicated in the `<foldout>` tag; in the above example either

```
<foldout value="10" />
```

or

```
<foldout value="11" />
```

would work.

7.8. An example

This is an example of a simple, but non-trivial, working parameter file:

```
<?xml version="1.0" encoding="UTF-8"?>
<jPDFImposition>
      <input value="ch01.pdf" pageNoOffset="-2" />
      <input value="ch02.pdf" />
      <input value="ch03.pdf" />
      <output value="book_bkl.pdf" />
      <format value="booklet" />
      <sheetsPerSign value="4" />
      <foldout value="5" />
</jPDFImposition>
```