jPDFImposition Manual

Version 0.5 – May 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.

It 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 their original relationship.

   This is all: the programme is ready to run.

4. 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 larger pages to be folded (and bound) together is called a *signature*.

4.1. *Pages* and *Formes*

As seen above, when dealing with impositions, we deal with two different kinds of pages:

- the original pages, the pages of the source document; in the following, these pages will be called **source pages** or simply **pages**.
- the 'pages' of the sheets to be printed once the imposition is applied; these 'pages' are usually larger, each containing several source pages. These 'pages' will be called with the traditional typographic term of **formes**.

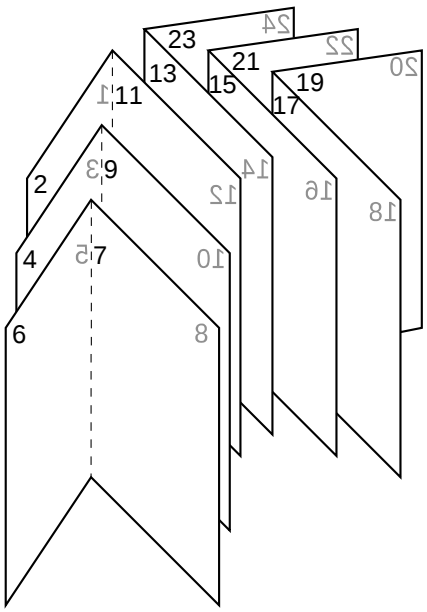
Each printable sheets is made of two formes: the *front forme*, the one containing the lowest numbered page, and the *back forme*, the other one.

4.2. Supported impositions

jPDFImposition knows about two main kinds of imposition:

Booklet:

Booklet formes are twice as wide as the source pages and each contains two source pages; a sheet (with its front and back formes) 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:

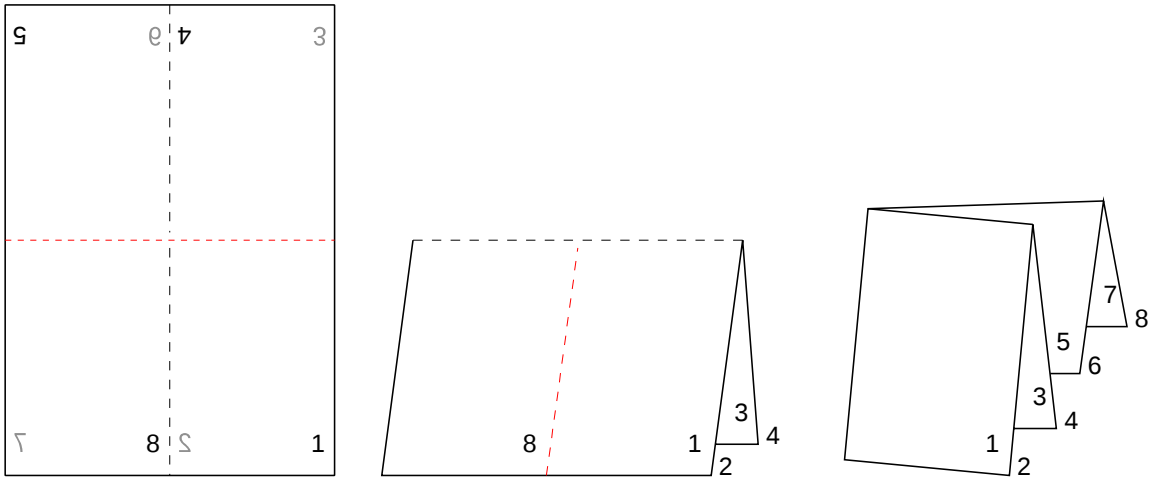


Standard impositions:

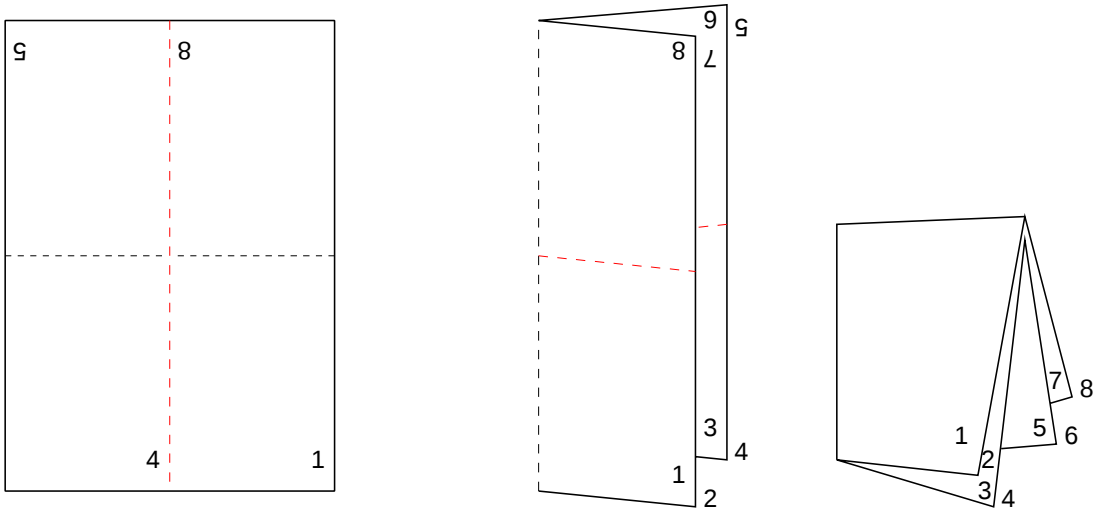
Traditionally, impositions were made out of rather large sheets, folded several times, resulting in many pages per sheet; in this kind of imposition, signatures are always made of just one sheet, with as many pages as come out of one sheet.

With two folds, each sheet forme yields four pages, this is traditionally called *in 4°*, three folds yield 8 pages per forme (*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.

5. Running *jPDFImposition*

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

TO BE DONE

6. Command line parameters

Running the programme by its name alone results in displaying a short help message. To do anything real, command line parameters have to be added.

The programme accepts the following command line parameters:

`-f <format name>`	One of: `booklet`, `in4h`, `in4v`, `in8h`, `in8v`, `in16h`, `in16h` or `none`.
`-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.
`-i <file name>`	The file name of the input PDF document to read from.
`-o <file name>`	The file name of the resulting PDF to create.
`-h`	Displays a summary of options and quits.
`-v`	Displays the version and quits.

A typical command line may look like:

```
jPDFImposition -f booklet -s 5 -i my-doc.pdf -o my-doc-bklt.pdf
```

or

```
jPDFImposition -l my-booklet-parameters.xml
```

7. Parameter file

The parameter list option (`-l parameter_file.xml`) tells the programme to read its parameters from the file specified, in addition to the parameters directly given in the command line.

The parameter file can give all the command line parameters described above and others, which is not possible to specify otherwise. The parameter file is intended as the primary way to control *jPDFImposition*.

If a parameter is given *both* in the command line and in the parameter file, the value given in the parameter file takes precedence.

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.

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. So, in the above example, the file `myfile.PDF` will be looked for in the same folder where the parameter file itself is.

7.2.1. `pageNoOffset` attribute

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 can be used:

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

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" />
```

Using these tags allows to directly give printed page numbers in other tags, instead of calculating the position of a certain page in the file page sequence.

7.2.2. `fromPage` and `toPage` attributes

These attributes are used to limit a source PDF to a range of pages only. Example:

```
<input value="myfile.pdf" fromPage="4" toPage="7" />
```

uses only 4 pages of `myfile.pdf`, from page number 4 to page number 7.

Note that these attributes use printed page number (in other words, they make use of the `pageNoOffset` attribute if given; if `myfile.pdf` has two unnumbered pages at the beginning, this tag:

```
<input value="myfile.pdf" pageNoOffset="2" fromPage="4" toPage="7" />
```

still uses only 4 pages, from page number 4 to page number 7, taking into account that they are actually the 6th to 9th pages of the document.

7.2.3. `signatureBreak` attribute

When this attribute is set, the input file will be the last of its signature and the next input file will start with a new signature. This is occasionally useful to include in the same output file contents which is intended to be printed separately. Example:

```
<input value="cover.pdf" signatureBreak="true" />
<input value="contents.pdf" />
```

the `cover.pdf file` will be output in its own separate signature, without affecting the imposition of the `contents.pdf` file.

An attribute value of `"true"` or `"yes"` or `"1"` sets the attribute; any other value turns resets it (i.e. has no effect).

7.3. <append> tag

Similar to the `<input>` tag, also defines a source PDF file; however, all the `<append>` files will be copied verbatim after all the `<input>` files, without any imposition applied. This may be useful to include in the output file descriptive or instructional matters which are not intended to be printed, like printing instructions or similar. Example:

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

Several `<append>` tags can be used to append several non-imposed.

As for the `<input>` tag, 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. So, in the above example, the file `myfile.PDF` will be looked for in the same folder where the parameter file itself is.

7.3.1. `fromPage` and `toPage` attributes

These attributes works in the same way as for the `<input>` tag.

7.4. <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.

7.5. <format> tag

Defines the imposition format. Example:

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

The possible values are:

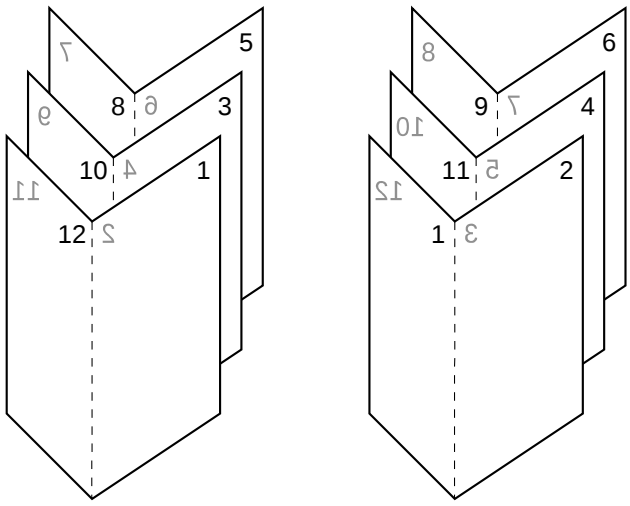
- `booklet`: the booklet format
- `in4h`: *in 4°*, horizontal first
- `in4v`: *in 4°*, vertical first
- `in8h`: *in 8°*, horizontal first
- `in8v`: *in 8°*, vertical first (rather unusual)
- `in16h`: *in 16°*, horizontal first
- `in16v`: *in 16°*, vertical first
- `none`: to copy the input files into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

Any other value defaults to the `booklet` format.

7.5.1. `firstPageAsEven` attribute

The `booklet` format accepts an additional attribute `firstPageAsEven`. Normally, the first page of a book is considered an odd page and goes to the right side of the first forme, its corresponding left side being occupied by the last page of that signature.

Occasionally, mainly in leaflets, it is necessary or convenient to place the first page on the left side, immediately followed by the second page, in order to have a double page on the outside of the signature. This sketch illustrates the difference:



Standard booklet Booklet with first page as even

The `firstPageAsEven` attribute does exactly this. Example:

```
<format value="booklet" firstPageAsEven="true" />
```

An attribute value of `"true"` or `"yes"` or `"1"` turns the attribute on; any other value turns it off. By default, this attribute is off.

7.6. <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" />
```

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

These parameters allow to displace the front and back formes of the resulting sheets 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 forme of all the sheets by 1.5 mm to the right (positive X) and 2 mm up (positive Y).

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

7.8. <pageSizeHoriz>, <pageSizeVert> tags

By default, *jPDFImposition* keeps the size of each source page as it is and sizes each forme width and height to be a multiple of the size of the *first* forme page.

If the source pages are not all of the same size, in either dimension or both, it is possible to force a width and/or a height for all the source pages, with the `<pageSizeHoriz>` and `<pageSizeVert>` tags.

They refer to the size of the *source pages*, not of the resulting *formes*; *jPDFImposition* will compute the forme width and height as appropriate multiples of the given page size(s).

Note that pages themselves are **not** scaled in any way; they are simply moved to be in the centre of the given common page size.

The tag values are in mm.

An example:

```
<pageSizeHoriz value="210">
<pageSizeVert value="297">
```

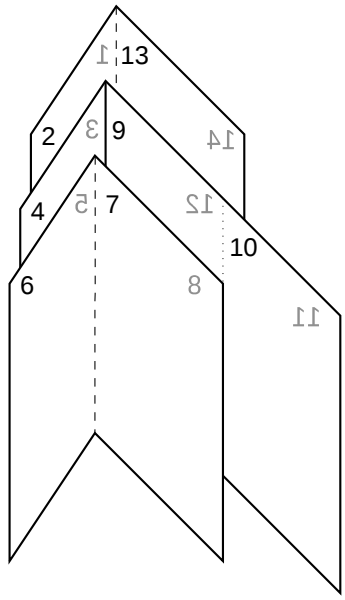
forces all the source pages to the A4 size (21 x 29.7 cm).

Technical note: the actual page size used to compute the forme size and/or the page centre comes from the PDF page value defined as *Media Box*, which is the size of the physical support on which the source page was intended to be printed. The page actual content may be (and usually is) smaller than this box; it might also be larger, but this usually implies the PDF is not correct (or has a very peculiar goal).

In no case, *jPDFImposition* scans or renders the page contents to ascertain the actual occupied area.

7.9. <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.

Pages 9 and 12 are shifted by 7 mm to the right to allow some room for the gluing.

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 will receive a small mark telling the user to which other page it is intended to be glued (in the above example, the text “p. 9” will be added in small print on the right edge of p. 4).

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.

7.10. An example

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

```
<?xml version="1.0" encoding="UTF-8"?>
<jPDFImposition>
<!-- 3 input documents; the first has two unnumbered pages at the
beginning -->
     <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" />
<!-- use pages slightly larger than A4 -->
     <pageSizeHoriz value="220" />
     <pageSizeVert value="307" />
<!-- one fold-out leaf with pages no. 5 and no. 6 -->
     <foldout value="5" />
```

```
</jPDFImposition>
```