So you want to replace your WYSIWYG word processor but 508 compliance has you stumped

Andy Clifton

Monday 16th September, 2013

Abstract

Ease of use and development of standard processes mean that WYSIWYG word processors are frequently used within organizations to produce documents that are edited and reviewed using tools built in to the word processors. Unfortunately, there are no widely-adopted methods of editing and reviewing LaTeX. Also, these WYSIWYG word processors can be used to produce PDF documents that pass automated accessibility testing. Although LaTeXproduces high-quality PDF files, these do not pass automated accessibility tests. There is therefore a need to produce LaTeX documents that can be transferred to WYSIWYG tools for review and editing, and where the output PDF satisfies accessibility requirements and tests.

This document explains how LaTeX can be used to prepare LaTeXsource that can be converted in to an *.rtf* file to be read in to a WYSIWYG word processor. The same source can be used to create *.pdf* files that satisfy accessibility requirements. This document can also be used as a template as it contains most of the elements of a technical LaTeX document, including complex document structures, lists, equations, figures and code listings.

Table of Contents

List of Figures

List of Tables

1 Introduction

The de-facto standard for scientific publishing is LaTeX. LaTeX is often preferred for technical documents because of the relatively simple file format that can be shared across users on many different platforms, and the ease of formatting a document for journal publication. In contrast, many organizations that have a well-defined publishing workflow are built around a small number of WYSIWYG word-processing tools, many of which are now integrated with PDF creators. Commercial word processing tools often include the ability to edit a document and track input from co-workers or dedicated editors. While changes can be compared between LaTeXdocuments using command line tools, many editors lack the skills to edit a LaTeX source file. How, then, can LaTeX be used in an collaboration with WYSIWYG word processing tools?

Another issue with using LaTeXfor publications is accessibility. This is particularly challenging for federally-funded organizations: since the US Congress passed the 1998 Section 508 Amendment to the Rehabilitation

Act of 1973, it has been a requirement that all federally-funded documents are accessible to people with disabilities. An accessible PDF has several characteristics:

- All of the document content has been tagged
- It is possible to define a reading order based on those tags
- Images and links are given alternate text descriptions
- Tables are tagged, so that the table structure can be established
- Unicode descriptions of all characters are required

A document that has these characteristics is often referred to as being '508 compliant'. This adds another level of complexity: not only does a document need to be converted into something that a WYSIWYG word processor can read, but the output *.pdf* needs to be 508-compliant. As 508-compliance is often judged using automated tests on the *.pdf* file, there is no option to work around this by using careful text descriptions of figures, for example.

In this document, we explain how a custom class file can be used to generate a range of document formats with common formatting, and how the document content can be shared with coworkers using WYSIWYG tools. We explain how the same source file can be used to generate a 508-compliant **.pdf* direct from LaTeX. We also show some of the limitations of the method and what remains to be fixed. Our goal is that this will be a 'living' document that can be updated as we gain new insight into this process.

2 A workflow for producing accessible .pdf files

- Define a class file that contains the correct formatting, etc, using article, report or book classes. Include the minimum number of up-to-date packages in the class and add the nag package to make sure that all users can see that those packages are not deprecated.
- Provide tools to add accessibility support to the *.pdf* file
- Create a template showing how to use the class file
- Create a source code repository for the class and template files, and distribute the URL of the repository to LaTeX users
- Create documents using the standard class file on local or network drives, or using collaborative tools like writeLaTeX
- Convert the tex files to rtf using LaTeX2rtf, available at http://latex2rtf.sourceforge.net
- Get edits and peer reviews done on a cleaned-up .rtf file that has been saved as .doc or .docx
- Transfer edits from the .doc or .docx document back in to 'tex manually, and complete the PDF production in LaTeX.

The rest of this section gives details on each of the above steps.

2.1 A "meta" LaTeX style file

A class file called *meta.cls* has been written to implement common formatting requirements in LaTeX across the standard LaTeX article, report, book, and memoir classes. The class file calls a common group of packages (Table 1) regardless of the chosen class. Slightly different options are chosen for those packages if the document uses the memoir class, or if the document uses chapters or not. The class can be customized to meet specific formatting requirements.

This approach gives a single file which contains all of the formatting required to produce a range of documents. For example, an organization that wants to implement an in-house LaTeX publishing system can customize the *meta.cls* and then make the class file available through a subversion server from which authors pick up the current class file. Each user can then install the class file in their tex structure or in the directory with their files. Similarly, changes are easy to implement in the style file and roll out to users, and because documents are LaTeX-based it is easy to recompile old documents.

The ***meta*** class is written so that it will accept all of the standard global LaTeX options, as well as options that are specific to the underlying class that the user has chosen. In the preamble, the user just has to write:

```
\documentclass[draft, report]{meta}
```

to specify the options (inside the square brackets) that will be passed to the ***meta*** class.

Options that have been implemented include:

book compile the document using the LaTeX *book* class. This is intended for longer documents and allows the use of chapters.

report compile the document using the LaTeX *report* class. This is intended for longer documents and allows the use of chapters.

article compile the document using the LaTeX *article* class. This is intended for shorter documents such as journal articles. This class does not support the use of chapters.

memoir compile the document using the LaTeX *memoir* class. This option is not recommended because of the challenge with later converting to rtf format.

draft add a 'draft' watermark to all pages and colours all links in blue.

10pt, 12pt set the font size accordingly. The default is 12 point.

letterpaper, a4paper set the paper size. the default is letter paper.

Table 1 lists the packages that are explicitly called by ***meta*** in the order they are called in. These packages often call other packages, so this is not an exhaustive list.

2.2 Accessibility support

LaTeX does not prepare a structured PDF document directly. Instead, we use the ***accessibility*** package to do this for us. This generates a tagged PDF that passes most automated document tests. The ***accessibility.sty*** package can be downloaded from http://www.babs.gmxhome.de/download/da_pdftex/accessibility.sty. Note that line 131 in ***accessibility.sty*** should be commented out (*pdfoptionpdfminorversion* is set elsewhere).

2.2.1 Alternative text

Alernative text, or 'Alt text', is a textual description of an equation, link or figure that can be used to replace the visual information in that element. This is often seen a text 'pop-up' in PDF readers. For example, passing the pointer over the following readers should reveal a pop-up:

$$a^2 + b^2 = c^2 \tag{1}$$

Alt text can be added after the PDF is compiled using a PDF editor such as Adobe's Acrobat Pro. Alternatively – and probably best for ensuring that the final document is what the author intended – it can be generated from within the source document using the **pdftooltip** environment from the ***pdfcomment*** package. The general form of the command is:

```
\pdftooltip{<item>}{<alt text>}
```

The previous equation was generated using this code:

```
\begin{equation}
\pdftooltip{a^2+b^2=c^2}{An equation}
\end{equation}
```

The same approach can be used to create alt text for images. For example, Figure 1 has been labeled.

2.2.2 Problems with embedded fonts

One requirement of passing automated tests for accessibility is that fonts must be embedded in the the final PDF. You can check the PDF for embedded fonts using a PDF viewer. For example, in Adobe Acrobat Reader, look at the 'fonts' tag of the document properties. If any fonts are not shown as being an *embedded subset*, you need to try again.

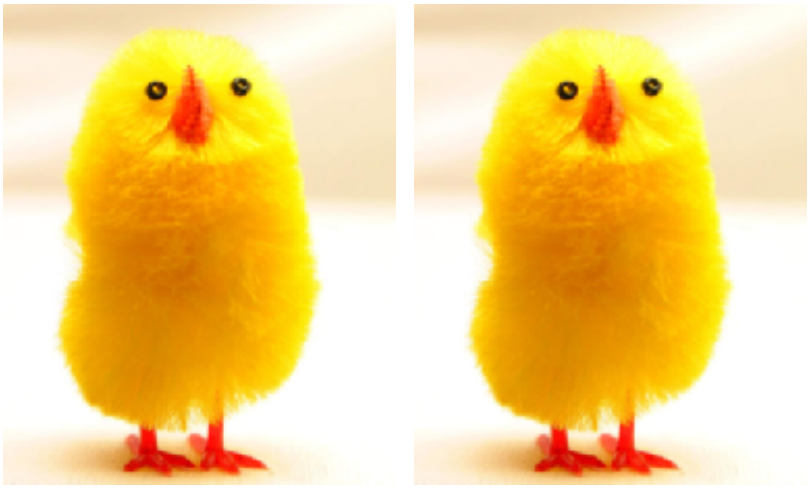
Encapsulated postscript figures are particularly prone to having undefined fonts. Check by compiling your document in draft mode, and seeing if the fonts are still present in the output PDF. To fix this problem, you could consider changing the *.eps* file to a *.png*. If you wish to do this 'on the fly', you could use this approach in your preamble:

```
\usepackage{epstopdf}
\epstopdfDeclareGraphicsRule
 {.eps}{png}{.png}{convert
eps:\SourceFile.\SourceExt
png:\OutputFile}
\AppendGraphicsExtensions{.png}
```

2.3 A template

Table 1. Packages explicitly loaded by the *meta* class. Unless otherwise stated, packages are not supported by `latex2rtf`.

Packages	options	functionality	`latex2rtf` support	`latex2rtf` workaround
nag		checks that packages are up to date and looks for bad habits in LaTeX code.		
geometry		sets page size and margins	✓	
mathptmx		changes fonts		
helvet		changes fonts		
courier		changes fonts		
amsfonts, amssymb		supplies fonts that are useful for mathematics		
booktabs				
graphicx		graphics handling, including *.eps* figures (see Section 2.2.1)	✓	
natbib	sort	handles citations and allows the `\cite`, `\citep` and `\citet` citation commands.	✓	
fontenc	T1			
xcolor				
babel	english			
subfig		provides the `subfloat` environment to produce sub figures	✓	*subfloat* is mapped to the *subfigure* command
hyphenat				
setspace				
parskip				
toclof	subfigure			
toclifbind	nottoc, notlot, notlof			
todonotes		inline and margin to-do notes	✓	Notes are prefaced with **To Do:** in the output
caption				
pdfcomment		tool-tips. Also calls the package ***hyperref***	✓	the tool tip is suppressed



(a) A chick. (b) Another chick.

Figure 1. Test images

The code used to produce this document is available from https://github.com/AndyClifton/AccessibleMetaClass.

2.4 Changing to a WYSIWYG-readable format using Latex2rtf

latex2rtf reads *.tex* files and converts common LaTeXcommands into their rich text format equivalent. It can be downloaded from `http://latex2rtf.sourceforge.net`.

We want to use the same source file for the PDF and for the conversion to a WYSIWYG-readable format. To ensure compatibility between the *.tex* file and ***latex2rtf***, we limit the packages used in the ***meta*** class to those that can be understood by ***latex2rtf***. Also, ***latex2rtf*** can read **`input`** files and has some capacity to deal with **`\newcommand`**s. Therefore, we have created a file to remap all of the environments that are available using the ***meta*** class to simpler forms that can be understood by ***latex2rtf***. This file is called *Latex2rtf_meta_with_acc.tex*.

If you are using the command line version of Latex2rtf, you can convert your latex file using the commands,

```
latex foo.tex
latex foo.tex
latex2rtf foo
```

You may get some error messages, which may or may not be important. Check the file was produced by opening the *.rtf* file in your favorite WYSIWYG package. If it works, save it in that proprietary format, and distribute it for your non-LaTeXing coworkers.

The following are suggested as best practices when working with ***latex2rtf***:

Convert early and often. Check that your document converts using ***latex2rtf*** every time you use a new environment.

Check new packages. If you add a new package to the *meta.cls* file (not recommended), try the conversion immediately, and then again once you actually use one of the new environments or commands. If you've added a new package that ***latex2rtf*** doesn't support, you may need to edit the file *Latex2rtf_meta_with_acc.tex* to redefine those commands to something that will get you through the conversion (and review).

Avoid custom commands. ***latex2rtf*** does not work well with custom commands. A list of all recognized commands is available in the manual at `http://latex2rtf.sourceforge.net/latex2rtf.pdf`. If you have custom commands, you may need to redefine those to work with the commands that ***latex2rtf*** does recognize, and place these in the preamble. To separate your commands from those that are required for compatibility with

the ***meta*** class, place your commands in a separate file (e.g *Latex2rtf_meta_with_acc_CUSTOM.tex*).

Remove all of your to-do notes. We've all forgotten to do this at one time or another...

3 Unified source document

So far we have identified two discrete paths that our document will take. These are

1. Compiling the LaTeX document using normal ***pdftex***
2. Converting to something portable using ***latex2rtf***.

And, to keep things tidy, we would like to use the same source for each step.

Fortunately, ***latex2rtf*** offers a boolean, **`\iflatextortf`**. If ***latex2rtf*** is used, **`\iflatextortf`** will be TRUE. This means that we can create a preamble that accommodates both paths:

```
\newif\iflatextortf

\iflatextortf
    \documentclass[10pt,
letterpaper]{report}
    \input{Latex2rtf_meta_with_acc.tex}
\else
    \documentclass[draft, report]{meta}
\fi
```

3.1 Including code listings

The ***listings*** package is one of several packages that can be used to typeset source code, and is used in this document. Unfortunately, ***listings*** is not compatible with ***latex2rtf***, and so we need a workaround that will be available to both ***latex2rtf*** and ***pdflatex***.

Using an example from Werner posted on tex.stackexchange.com, we default to the **`verbatim`** environment, and only use ***listings*** if we are compiling with ***pdflatex***:

```
\makeatletter
\@ifundefined{lstlisting}{}{%
  \let\verbatim\relax%
  \lstnewenvironment{verbatim}{
      \lstset{language=[LaTeX]TeX,
      ... % here we can customize the
          listing
  }}{}%
}
\makeatother
```

3.2 The final preamble

```
\newif\iflatextortf

\iflatextortf
    \documentclass[10pt,
letterpaper]{report}
    \input{Latex2rtf_meta_with_acc.tex}
\else
    \documentclass[draft, report]{meta}
\fi

\makeatletter
\@ifundefined{lstlisting}{}{%
  \let\verbatim\relax%
  \lstnewenvironment{verbatim}{
      \lstset{language=[LaTeX]TeX,
      ... % here we can customize the
          listing
  }}{}%
}
\makeatother

\author{Andy Clifton}
\title{So you want to replace your
WYSIWYG word processor but 508 compliance
has you stumped}

\begin{document}
```

4 Problems with this approach

There are a few things that still need to be fixed.

- Problems associated with the ***accessibility*** package:
  - formatting in section headings such as `\emph{}` breaks compiling
  - empty *.bbl* files also break it
  - there's an extra line at the start of **`lstlisting`** environments

5 Conclusions

In order to generate accessible *.pdf* files with LaTeX that can also be shared with WYSIWYG word processor users, a LaTeX meta class was created. This meta class uses a well-defined set of packages that limits the scope of the casual 'tex author from fiddling, whilst giving them tools to create complex technical documents. Accessible *.pdf* files are compiled directly from the LaTeXsource code, while *.rtf* files are written using the ***latex2rtf*** program. This meets the users' need for a single source file and minimal intervention.

Acknowledgements

This document benefitted from contributions to the website, `http://tex.stackexchange.com/`.