Report on the LaTeX Tagged PDF workshop, TUG 2023

David Carlisle, Ulrike Fischer,
Frank Mittelbach

Contents

Introduction

On the afternoon before the formal conference program, the LaTeX project held a workshop, led by Ulrike Fischer, on generating tagged PDF from LaTeX. The workshop was well attended with more than thirty people participating — a good mix of package developers and end users. We thank DANTE e.V. for very generous financial support.

The workshop was split into three parts. Firstly, a general introduction to tagging in PDF. Secondly, a demonstration of the process that a class or package maintainer should take to modify the code to produce well-tagged PDF. The `acmart` class was used for the example as its author, Boris Veytsman, was attending the workshop. Finally, we had a more open discussion on issues and desired syntax for structured tables.

1 Tagged PDF

Ulrike gave a brief overview of how PDF tagging encodes the structure of a document in a PDF file and why with is important not "just" for accessibility requirements. Readers are encouraged to read the introduction to the `tagpdf` package for a similar, more detailed exposition.

1.1 Engine considerations

The recommended engine for tagging is luaLaTeX. For legacy documents pdfLaTeX is supported, but it has some small problems, for example, it sometimes doesn't insert real space characters in places where they are needed. It also requires more compilations to build the correct tagging structure.

Other workflows such XeLaTeX or LaTeX–`dvips` are not recommended as real space characters can't be inserted in these cases. In order for accessibility tools to distinguish inter-word spaces from inter-letter kerns and other spacing adjustments, words need to be separated by space characters (U+0020) even if the spacing is further adjusted. It is not feasible to add these space characters just using the macro layer, and currently only pdfLaTeX and luaLaTeX have engine-level support to add them.

When developing or updating packages or classes for tagging, one always needs to test tagging with at least pdfLaTeX *and* luaLaTeX. They use different methods to create the basic MC-chunks (called "marked-content sequences" in PDF reference manuals) and create the structure tree (namely, PDF literals in pdfTeX and luaTeX node attributes in luaTeX).

1.2 Tagging commands and Tagging activation

The `tagpdf-base` package provides dummy versions of all important tagging commands. It is loaded automatically by `\DocumentMetadata` but can also be loaded by other packages.

The `tagpdf` package is loaded (and tagging is then automatically activated) by using a `phase` key:

```
\DocumentMetadata{testphase=phase-III}
```

It is possible to produce untagged PDFs with the new code in `latex-lab` (both for the final version or in draft mode to speed up compilation), but a suitable interface is currently missing. It will be added later.

1.3 Compatibility with older formats and legacy code

The `latex-lab` code and tagging in general require a current LaTeX or even `latex-dev`. If necessary, a class or package can test the format version with `\IfFormatAtLeastTF` or `\@ifl@t@r\fmtversion` to provide fallbacks for older formats.

Whether a document uses `\DocumentMetadata` can be tested with `\IfDocumentMetadataTF`. With `\tag_if_active:TF` it is possible to test if tagging is active.

At the moment tagging can only be activated through the testphase keys in `\DocumentMetadata`, which cannot be used in a class because this command must come before `\documentclass`. This means that at this point in time a class cannot trigger tagging — during the test phase this is the decision of

doi.org/10.47397/tb/44-2/tb137carlisle-taggedpdfworkshop23

the author of the document. A class can only check the state and issue an error or a warning if tagging is not activated.

1.4 Tools

To check the tagging structure you need a tool, or several. Some possible options:

- Adobe Acrobat Pro (non-free, `adobe.com`)
- PDF XChange Editor (a free version is available, `pdf-xchange.eu`)
- PDFix (free Desktop Lite version is enough, `www.pdfix.net`)
- PAC 2021 (a checker, `pdfua.foundation/de/pac-2021-der-kostenlose-pdf-accessibility-checker`). Currently it doesn't work with PDF 2.0.
- Big Faceless PDF Library (free trial version, `bfo.com/download`). A Java library that can also be used to dump information into text files.
- RUPS (`itextpdf.com/products/rups`)
- An online service you can try: `ngpdf.com`

2 Updating a LaTeX class

In the second part of the workshop the general principles of how to update a package to be compatible with PDF tagging were discussed, illustrated with specific examples from `acmart` class.

2.1 Adapting packages and classes

Broadly, packages and classes have to handle two problems:

- They often redefine internal LaTeX commands and environments (directly or through a package or class they load) and this breaks the tagging support provided by `latex-lab`.
- Any new commands and environments defined by the package need to be adjusted to add or correct the tagging.

Some general recommendations on strategies to address these issues:

- For all existing redefinitions of LaTeX's core commands, check their purpose and consider if the redefinition can be avoided, e.g., by making using of the recently-introduced hooks in various core commands of LaTeX for precisely this reason, or by making a feature request to enhance the LaTeX command to support your use case directly or via new hooks.
- New commands and environments built on top of existing commands can inherit the tagging support from the kernel. For example, bibliographies are typically simply lists. Check if the resulting structure is ok.
- Paragraphs are automatically tagged through the paragraph hooks. Not every structure is allowed inside a paragraph. It is therefore important to check how your own commands behave both in horizontal and vertical modes.
- Complex commands can be difficult to tag correctly. For a first draft it is possible to put a minimal structure around them and then to stop tagging. As an example, a complicated `\maketitle` command could be handled as follows, i.e., by disabling tagging for the title:

```
\AddToHook{cmd/maketitle/before}
  {\tagstructbegin{tag=Title}%
   \tagmcbegin{}\tagstop }
\AddToHook{cmd/maketitle/after}
  {\tagstart\tagmcend\tagstructend}
```

Participants were, and readers of this report are, encouraged to give feedback. The tagging support is still in development. If something doesn't work as expected or is too complicated we need to know about it. Comments can be added as issues or discussions at `github.com/latex3/tagging-project`.

2.2 `acmart`

The `acmart` class is used to typeset articles for the Association for Computing Machinery. It is around 3500 lines of code, based on `amsart`, and supports various journal styles.

The aim of the workshop was not to fully update the entire class to be tagging-aware, but to show the steps needed for this, and to show how relatively simple changes can already significantly improve the automatic tagging in documents, even when generated by large production journal code.

2.2.1 Review of packages and commands

The initial step was to generate a list of all packages used by the class, around 40 packages in this case. Some are known not to affect tagging, some (notably those affecting footnotes such as `manyfoot` and `nccfoots`) may affect footnote tagging. They would need to be checked in a final version; specifically, whether newer versions of the packages are tagging-aware or whether there is "first-aid" support for the package in the `latex-lab` code.

The main issue addressed in this session was classes that copy "original" definitions of standard LaTeX commands, but often they copy older versions and so are missing updates, in particular the (relatively) recent changes to add LaTeX hooks and tagging support.

For example, in the case of `acmart`, it intends to simply restore standard LaTeX section headings (undoing the changes made by the underlying `amsart` class). It does this by redefining `\@startsection` and related commands using copies from an older LaTeX format. This has the effect of undoing the additions to support tagging section headings added by the tagging code.

In this case, a simple fix would be to modify the class to save the definitions (for example with `\DeclareCommandCopy`) before loading `amsart` and restore them afterwards. This has the effect of restoring the tagging-aware section commands, if they were activated by `\DocumentMetadata`.

Pointers were given to changes that would be required in other parts of the class. Tables of contents would need changes equivalent to the changes made for tagging to the standard `\@dottedtocline` and `\@starttoc` commands. The class has quite extensive code modifying footnotes and this must be checked for tagging. Perhaps in initial versions, tagging would be disabled for footnotes.

As noted above, the title page handling (as is common in journal classes) is rather complicated and would need some custom tagging support, but can in initial versions be easily customized not to tag, so that no invalid tag structures are generated.

Similarly, the class has redefinitions of `minipage` that would conflict with minipage tagging and need to be checked.

In the workshop there was only time to look at the redefinitions of standard commands made by the class, and how to recover tagging support based on the support implemented for the standard definitions. What was not addressed in the session — but would be needed for a fully tagging aware class file — would be to add appropriate tagging support to any new commands and environments defined by the class. This requires understanding of not only the TeX coding details but also, and more importantly, an understanding of the intended structure implied by the commands; that is, what tagged structure would be desirable in each case.

Going forward, this raises an important point for class and package maintainers. To produce well-tagged documents it is not only important to produce a pleasing visual layout for new constructs, it is equally important to think about what structures are represented, and how that structure should be tagged. For example, "misusing" other elements for purely visual effects is likely to produce completely inadequate structural results.

3 Tables

The final part of the workshop involved a lively discussion of markup for tables. It is already possible to use the low-level tagging commands from `tagpdf` to tag tables and individual cells, although the current test phase does not have code to automatically infer table structure.

The main problem is that classic LaTeX `tabular` markup does not distinguish table headers or any other structural features, it is purely concerned with the visual layout.

This leads to two related issues:

1. What would be a good table markup for new documents?
2. What heuristics could or should be used to infer table structure for the existing corpus of LaTeX documents?

For the first issue, markup used by LaTeX table packages were considered, also ConTeXt table markup, and the table markup in HTML (which is very close to the final required structured tagging in PDF).

For the second issue, various possibilities were discussed ranging from a view that it is better never to "guess" and that no table headings should be inferred, through a simple heuristic such as always assuming the first row is a heading, to more detailed heuristics looking at fonts and/or position of `\toprule` from `booktabs` or `\endhead` from `longtable`.

This is still very much an area of ongoing activity for the LaTeX team, and people are invited to contribute to the discussion which is now online in the GitHub discussion site set up for the tagging project mentioned earlier: `github.com/latex3/tagging-project/discussions/1` has some thoughts from workshop participants on the table syntax. Feel free to join the party with further ideas.

◇ David Carlisle
LaTeX project team
Oxford, UK
`david.carlisle (at) latex-project.org`

◇ Ulrike Fischer
LaTeX project team
Bonn, Germany
`ulrike.fischer (at) latex-project.org`

◇ Frank Mittelbach
LaTeX project team
Mainz, Germany
`frank.mittelbach (at) latex-project.org`