Automatically producing accessible and reusable PDFs with LaTeX

Frank Mittelbach
frank.mittelbach@latex-project.org
LaTeX Project
Mainz Germany

Ulrike Fischer
fischer@troubleshooting-tex.de
LaTeX Project
Bonn Germany

David Carlisle
d.p.carlisle@gmail.com
LaTeX Project
Oxford UK

Joseph Wright
joseph@texdev.net
LaTeX Project
Ely UK

Abstract

In this application note we outline the goals of the "LaTeX Tagged PDF" project, describe its current status, show how it can already now been used to create accessible and reusable PDFs, and outline our future plans for a successful completion. Further information can be found at https://latex3.github.io/tagging-project/.

CCS Concepts

• **Software and its engineering** → *Open source model*; • **Applied computing** → **Format and notation**; **Document metadata**; • **Human-centered computing** → **Accessibility technologies**; • **Information systems** → **Document structure**.

Keywords

Accessibility, PDF/UA, Well Tagged PDF, LaTeX, Typesetting systems, Tagging, Reuse

ACM Reference Format:
Frank Mittelbach, Ulrike Fischer, David Carlisle, and Joseph Wright. 2024. Automatically producing accessible and reusable PDFs with LaTeX. In *ACM Symposium on Document Engineering 2024 (DocEng '24), August 20–23, 2024, San Jose, CA, USA*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3685650.3685670

1 Introduction

For more than three decades now, the LaTeX system has been used, widely and successfully, for document production in the STEM world (Science, Technology, Engineering and Mathematics) and also in other places where high-quality output is required; but until recently its focus was solely on page-oriented output for print (on paper) or as paged output using the PDF format.

Nowadays, for many reasons, great interest has arisen in the production of PDF documents that are "accessible" and "reusable", in the sense that they contain information to assist screen reading software and data extraction, etc., and, more formally, that they adhere to the PDF/UA (Universal Accessibility) standards [1, 2] and the WTPDF (Well Tagged PDF) standard [8].

However, at present, all methods for producing such "accessible and reusable PDFs", including the use of LaTeX, require extensive manual labor[1] during either the preparation of the source or the post-processing of the PDF (maybe even at both stages); and these labors often have to be repeated after making even minimal changes to the (LaTeX or other) source.

To improve this situation for authors using the LaTeX system, we started a multi-year project to modernize and enhance LaTeX so that it becomes capable of "automatically" producing tagged and accessible PDF, adhering to standards, such as PDF/UA-2 and WTPDF (or, if necessary, to the older PDF/UA-1), without the need for any post-processing steps.

At the current time, a prototype version of the enhanced LaTeX system is already capable of generating such accessible and reusable PDFs, making it one of the first of the few PDF producers to date that can *automatically* produce documents according to the new PDF/UA-2 standard published in 2024.

2 The project setup

The project to enhance LaTeX to produce accessible and reusable PDF was initiated in 2020 with the production of a feasibility study [7], available from the LaTeX Project website [4]. The purpose of this forty-page document was to provide a high-level, but sufficiently detailed, understanding of the work and resources necessary when altering and enhancing the LaTeX system in a way that it becomes capable of automatically producing documents in which the semantic structure (that is available in the LaTeX source) is properly carried through to the output format, resulting in a tagged and accessible PDF. This initial study was addressed primarily to an audience within Adobe,[2] consisting of engineers and managers with a deep knowledge of digital typography and electronic publishing but not necessarily much background within the specialized world of TeX, LaTeX and its role in the accademic world. It therefore started with a longer overview of the benefits of the project and goes on to explain why LaTeX documents make a good starting point for the production of tagged and reusable PDF.

For many reasons, last but not least because of the complexity and diversity of the LaTeX software, the conclusion from the study was that this undertaking would require several years of

[1] If not using the already existing code extensions to LaTeX provided by the project.
[2] The public, cited, version of the study was updated in September 2020 with some minor redactions, corrections and clarifications.

DocEng '24, August 20–23, 2024, San Jose, CA, USA

ACM ISBN 979-8-4007-1169-5/24/08
https://doi.org/10.1145/3685650.3685670

concentrated work—an endeavor impossible to deliver by the small number of volunteer developers in their spare time (or at least not in a reasonable time frame).[3]

Fortunately, our Feasibility Study convinced the senior management of Adobe that this work would be a worthwhile project to improve the position of PDF in the STEM world and beyond, and they agreed to finance a major portion of the project work. At this point, however, Covid intervened and put a spanner in the works, and the initial commitment was delayed and stretched out so that it wasn't possible to keep the schedule, as outlined in the Feasibility Study. Nevertheless, the project got started, albeit at an initially slower pace.

2.1 The phases of the project

From the outset, the project was divided into six distinct phases which were to follow the typical LaTeX release cycles, i.e., each phase ends with a normal LaTeX maintenance release. These releases normally happen in spring (between May/June) and in fall (October/November), so typically in half year intervals. For these phases the work items were organized so that useful intermediate results become available as soon as possible in order to attract timely feedback and early adoption.[4] For example, after phase 2 it was already possible to automatically generate tagged PDFs for a restricted set of documents. In phase 3 the coverage was extended; with support for more constructs offered by extension packages. This is an ongoing process that will only finish with the conclusion of the project.

As a realistic scenario we expected that each phase would take between one and two release cycles, which meant that the overall project stretches across four to five years as a minimum, but probably somewhat more. Experiences with the first phases have shown that this was a fairly accurate prediction, somewhat slowed down by fluctuation in the financial support.

For the remaining phases additional funding will help to ensure timely delivery of the phase results and may allow the scope to be broadened in some areas, but as we already made clear in the Feasibility Study any expectation of earlier delivery would not be realistic given the complexity of the topic.

It is also important to note that all updates to important external packages are to be done using external resources, i.e., by the maintainers of those packages. This assumption is probably not valid in many cases (see discussion in Feasibility Study, task 2.4.3). In that case additional work has to be outsourced or undertaken as part of the project, and this will alter the timeline and funding requirements for phase 5 and 6, and possibly require a phase 7.

In the following we give a brief summary of the work carried out (or planned to be carried out) in the individual phases according to the Feasibility Study.

2.1.1 Phase I — Prepare the ground. The main goal of this phase was to provide the necessary basis for all the following phases. This included design and implementation of internal functionality, such as low-level manipulation of PDF objects or a proper hook mechanism, necessary to support later changes with minimal disruption for the existing user base. This phase finished in 2021/Q3.

2.1.2 Phase II — Provide tagging of simple documents. The main goal of this phase was to provide automatic tagging of simple documents, excluding more complicated structures such as mathematics, tables, etc. This involved setting up the necessary core code that provided the general mechanisms, dealing with the issues around automatic detection of paragraph text and tagging it, and by enabling a subset of the document elements to produce tags. This phase finished in 2022/Q4.

2.1.3 Phase III — Remove the workarounds needed for tagging. In this phase we extended the coverage of automatic tagging and removed several workarounds that had been necessary initially to provide a working prototype. For more complex structures such as math and tables, first trial implementations were devised. Both areas require active research that was initiated at that time.

During the work of phase 2 several new tasks were identified, e.g., the need for a configuration mechanism complementing the hook system and a universal key/value mechanism for use with most document-level environments and commands as an upward-compatible extension of LaTeX. These were added to this phase and the next, slightly increasing the timeline. This phase finished in 2023/Q4.

2.1.4 Phase IV — Make basic tagging and hyperlinking available. The original goal of this phase was to incorporate all the code currently in prototype packages into the kernel itself. However, during the project we decided on a more gradual approach and while we now provide an interface to access the code directly from standard LaTeX we nevertheless keep the tagging code for now in a `latex-lab` area. One reason for this change was the realization that the later phases would likely still need some adjustments to the underlying low-level code that could be more easily provided in a setup that supports multiple prototypes.

For the same reason the second main task of this phase, to provide support for hyperlinking directly in the LaTeX kernel was delayed, though initial support work was carried out in 2023/Q4.

Thus, we basically decided that this phase needs to be moved towards the end and instead immediately started working on tasks of the next phase.[5]

2.1.5 Phase V — Provide extended tagging capabilities. The plans for this phase focus on research, design and implementations for math and table tagging. Furthermore, interfaces for specifying alternate text are being developed and added to all relevant elements, such as graphical elements.

In addition, the coordination of updates to external packages was planned, to make their document elements "tagging enabled", as the core infrastructure for this was expected to be available as part of the LaTeX kernel.

2.1.6 Phase VI — Handle standards. The main goals of this phase were to provide support for the relevant PDF standards (as far as

[3] Despite the fact that millions of users in the STEM world and beyond rely on the quality results of LaTeX, the development and maintenance of its core has been undertaken by a handful of typography enthusiasts during the last three decades.

[4] Of course, during the various phases much will be 'work in progress' and so we expect testers and early adopters to work with understanding of such temporary limitations and of the possibility that extra installation steps, etc., will be necessary.

[5] This is a somewhat typical case for long-running projects initially planned in a waterfall model: eventually parts of the plan need adjustments to project reality and increased understanding of the dependencies.

this is possible using LaTeX without post-processing the resulting PDF), and adding kernel support for outlines and associated files.

Furthermore, a number of tasks started in previous phases, e.g., the coordination of updates to external packages, would finish then.

2.1.7 Research work. As outlined in the Feasibility Study, there are areas that will require original research prior to any implementation effort. The need for such work has become even clearer from current the discussions in the PDF Reuse TWG (a PDFA Technical Working Group, covering well tagged PDF [8]).

This research work forms a stream of activities parallel to the six phases and, while the original project plan shows most of this to be taking place during the later phases of the project, we have already started some of the ground work.

3 The current state of the project

As indicated in the previous section, the project team has now finished phases 1–3 and some aspects planned for the remaining phases. In particular this means that it is now possible to automatically generate accessible and reusable PDFs that adhere to the standards PDF/A-4f, PDF/UA (1 or 2), and WTPDF; as long as the LaTeX source only uses document elements described in the LaTeX manual [3] or those provided by a small (but growing) number of important extension packages, such as amsmath, array, biblatex, hyperref, or longtable to name a few.[6]

A varied collection of sample documents can be reviewed at the project GitHub repository [6].

3.1 How to make use of the prototype

The new code can be used with pdfLaTeX or the Unicode engine LuaLaTeX.[7] The latter is the preferred engine recommended for new documents.

To enable structured PDF output from a well-behaved LaTeX source one has to add a single `\DocumentMetadata` declaration at the very beginning, i.e., before the `\documentclass`. In there, one declares important meta data, such as the document language, the requested PDF version, the standard(s) that the document should comply with, etc. Depending on the source document, that may be all that is required.

If targeting an accessibility standard then one of the requirements is that graphics will either offer an alternative text (to be used by AT tools) or be marked as strictly presentational. This is achieved by using an `alt={...}` key on `\includegraphcs` or by marking it with an `artifact` key—this requires minor modification of the LaTeX source for existing or new documents that use graphics.

The following gives an example; note that the key `testphase` is only necessary for enabling specific prototype modules and will not be required when the project finishes:

```
\DocumentMetadata{
  lang        = de,
  pdfversion  = 2.0,
  pdfstandard = ua-2,
  pdfstandard = a-4f, %or a-4
  testphase   = {phase-III,title,table,math,firstaid}
}
\documentclass{...}
\begin{document}
  ...
  \includegraphics[alt={A yellow duck}]{duckimage}
  ...
  \includegraphics[artifact]{decoration}
  ...
\end{document}
```

So far, only rudimentary support for tables is provided, i.e., by default, all table cells are marked as data cells. There are possibilities to declare header rows for individual (or all) tables of a document, but the interfaces are temporary and not comprehensive. An appropriate interface design and its implementation is part of the ongoing development, see section 4.1.

Math formulas are handled by the current prototype without any user customization. Some details of the current facilities for math tagging and our plans for the future are given in section 4.2.

More details on the current interface capabilities and their use are given in the document "Using the prototype for accessible PDF" available from the project GitHub repository [5].

4 Ongoing and future work

With phase 4 being deliberately delayed, the team is now addressing tasks from phases 5 and 6, in particular developing interfaces for complex tables and developing best practices for handling math.

Even with the delay to most of phase 4, we have already added the necessary infrastructure to the LaTeX kernel to make it possible to update external packages and classes to become "tagging enabled".

One important part of the future work is therefore to support such updates by developing the right level of developer guidance and in general coordinate the effort. Depending on the outcome here, it will also become clear whether or not there has to be a phase 7 in which the team takes ownership of the upgrade work for important extension packages that could not be adjusted by their maintainers for one or the other reason.

4.1 Tables (tabulars)

Automatically tagging tables in LaTeX[8] is now possible for simple tables with a predictable structure: a clear definable number of header rows and little to no adjustment to the visual formatting. However, in real documents, tables take on a wide variety of structures and these need to be described correctly in tags for the data to be extractable.

The method by which tables are implemented in standard LaTeX structures makes this challenging: for efficiency reasons, the data are not all read into memory in one step. The standard LaTeX table input syntax is designed around this approach, but this makes some concepts difficult to express. Work on richer table tagging may therefore require extension or revision of core table input structures, at least in as far as tagging more complex structures is concerned. (Whilst complex tables must eventually be handled, the ability to tag existing tables in many cases will be sufficient.)

[6] In particular, this means that out of the box only the standard document classes are fully supported. Other document classes may or may not work without adjustments.

[7] XeLaTeX is not capable of adding real spaces in the output: a requirement of the PDF standards.

[8] In LaTeX, the `table` environment is simply a container of something which may be a table; the document environment which creates a row/column grid is called a `tabular`.

4.2 Math

PDF 1.7 (and consequentually PDF/UA-1) provides a single mechanism to tag each math formula with an "Alt" attribute with a textual description of the formula's content. PDF 2.0 and PDF/UA-2, however, offer much richer possibilities for tagging mathematics. Firstly the PDF 2.0 tags include tags from the MathML namespace allowing, in principle, direct tagging of the typeset math. Secondly, each math formula may be associated with one or more "Associated Files" which are embedded in the PDF. The project code supports associating formulae with both the original fragment of TEX source code, and a MathML document. At the current time the MathML version to be associated with each math fragment must be provided via an external file, but tools to generate this file using existing TEX to MathML converters such as TeX4ht or LaTeXML are in development. The project is also developing a new TEX to MathML converter implemented directly within the LuaLATEX code. This should avoid the need to use a separate conversion process.

MathML may provide an accessible reading of the formula, but as it is associated with the whole equation, it may not be possible for a screen reader to navigate easily to any subterm of the formula and start from that point. This could be addressed by directly tagging the typeset formula with tags from PDF2.0's MathML namespace, however producing such tagging while not affecting the visual layout of the mathematics is technically challenging and will be addressed in a later phase of the project.

4.3 Semantic structures (a.k.a. tags) in PDF

When PDF 1.3 introduced a structure tree into the format, to support the inclusion of the document's logical structure, it used only a fairly minimal set of structure tags that were largely modeled after the basic HTML tag set. For example, mathematical formulas had to be tagged as a whole with a single `<Formula>` tag, without a possibility to add further structure within the formula. But also in other areas the structures were fairly simple and incapable of capturing more complex semantic information necessary for good access through AT software or other reuse without applying heuristics that unfortunately differed from one tool to the next.

This is the tag structure on which the PDF/UA-1 standard is built upon and it is one of the reasons why more complex PDFs (e.g., STEM documents), even if complying to this accessibility standard, are generally perceived as largely inaccessible.

PDF 2.0 improved a lot on this by providing a much richer tag set, and the new PDF/UA-2 standard (based on PDF 2.0) is therefore much better suited to offer real accessibility—once it is generally supported by AT tools.[9] As the result of the project work, LATEX users are capable of producing many documents complying to the PDF/UA-2 standard, i.e., providing extended semantic information based on PDF 2.0 in the output.[10]

However, even the tag set supported by PDF 2.0 is still fairly limited when it comes to complex documents, e.g., those typically seen in academic texts in STEM or humanities. Thus, when preparing a PDF to be PDF/UA-2 compliant, compromises have to be made, and some of the important structural information is lost when you transform a LATEX source into a PDF document.[11]

As part of the project we are therefore developing an extended tag set that describes the semantic structure of (complex) documents in more granular detail; this will help PDF processors (such as viewers) that understand this tag set to make better use of a document's structure. Ideas from this development may also prove useful in conjunction with future HTML5 developments.

LATEX is an open system that allows for structural extensions (and even changes to structures) in every direction. It is therefore not possible to define a definitive document model that is both valid and comprehensive for each and every conceivable LATEX document. However, it is possible to define a document model which captures the majority of LATEX documents that are out there in the real world. Combined with methods to extend (and possibly alter) the document model whenever necessary for special structural extensions or changes, we are confident that a comprehensive solution can eventually be provided.

This tag set (called a namespace in PDF 2.0) will thus be noticeably more detailed and comprehensive than those offered by PDF 2.0 and HTML5. We are working with the PDF Association [9] and various application producers to ensure that this namespace will, when complete, become a recognized[12] resource; it may also be more generally useful as an XML schema. This will, for example, allow PDF and other applications to directly use the extended tag set it provides; and this will enable such applications to make better use of the information contained in the document, whether for accessibility support or for other purposes.

For applications that do not (yet) understand this new namespace, we provide role-mapping back into PDF 2.0 (or PDF 1.7) as necessary; but of course, in that case the more granular information provided by the tags in the new namespace will get at least partially lost.

References

[1] ISO 2014. *ISO 14289-1:2014* (2nd ed.). https://www.iso.org/standard/64599.html PDF/UA-1. https://www.iso.org/standard/64599.html.
[2] ISO 2024. *ISO/FDIS 14289-2* (1st ed.). https://www.iso.org/standard/82278.html
[3] Leslie Lamport. 1994. *LATEX: A Document Preparation System: User's Guide and Reference Manual* (2nd ed.). Addison Wesley.
[4] LATEX Project Team. [n. d.]. Website of the LATEX Project. https://latex-project.org/.
[5] LATEX Project Team. 2024. Using the prototype for accessible PDF. https://github.com/latex3/tagging-project/.
[6] LATEX Project Team. 2024. WTPDF / PDF/UA-2 Examples by the LATEX Project. https://github.com/latex3/tagging-project/discussions/72.
[7] Frank Mittelbach, Ulrike Fischer, and Chris Rowley. 2020. *LATEX Tagged PDF Feasibility Evaluation.* LATEX Project. https://latex-project.org/publications/indexbyyear/2020/.
[8] PDF Association 2024. *Well-Tagged PDF (WTPDF)* (1.0.0 ed.). PDF Association. https://pdfa.org/wp-content/uploads/2024/02/Well-Tagged-PDF-WTPDF-1.0.pdf
[9] PDF Association (PDFA). [n. d.]. Website of the PDF association. https://pdfa.org/.

Received 10 June 2024; accepted 12 July 2024

[9] There is the typical chicken and egg problem: for simple documents PDF/UA-1 was sufficient and for complex documents the production based on PDF 2.0 was difficult and required manual work by the authors so was basically not done. As a result, there was no incentive for tools to support it. With LATEX now becoming capable of automatically producing such documents the situation is already changing.

[10] Limited as detailed above to documents using the core LATEX structures, but with a clear roadmap to wider coverage.

[11] The same is also true if HTML documents are produced: they also cannot (correctly) express the semantics of many real-life documents. The reason that HTML documents are nevertheless considered more accessible in the community is due to the fact that the comparison is made with PDF 1.7 based documents (i.e., PDF/UA-1) and not with those based on PDF 2.0. It is also caused by the (so far) missing support in PDF viewers embedded in Web bowsers that typically do not support structured PDFs at all.

[12] Preferably acknowledged in future revisions of the PDF standard.