LaTeX Tagged PDF — A blueprint for a large project

Frank Mittelbach, Chris Rowley

Abstract

In Frank's talk at the TUG 2020 online conference we announced the start of a multi-year project to enhance LaTeX to fully and naturally support the creation of structured document formats, in particular the "tagged PDF" format as required by accessibility standards such as PDF/UA.

In this short article we outline the background to this project and some of its history so far. We then describe the major features of the project and the tasks involved, of which more details can be found in the Feasibility Study [8] that was prepared as the first part of our co-operation with Adobe.

This leads on to a description of how we plan to use the study as the basis for our work on the project and some details of our planned working methodologies, illustrated by what we have achieved so far and leading to a discussion of some of the obstacles we foresee.

Finally there is also a summary of recent, current and upcoming activities on and around the project.

Contents

1 A short history of the project 901

2 What the project covers 902

3 Explaining the blueprint 904
- 3.1 The major tasks 904
- 3.2 The project plan 904
- 3.3 Timeline and other considerations . . 906

4 First results 906

5 Outlook 906

1 A short history of the project

When TeX was developed by Don Knuth in the early 1980s, it was designed as a high-quality automated typesetting system that concerned itself solely with the production of a "printed result" from text input, with paper as the final intended output medium. Any other kind of typeset output was either not supported or not directly supported.

Partially in parallel to Knuth's work, Leslie Lamport developed LaTeX as a complex and highly integrated collection of TeX macros that "runs on top of TeX". Some time later, during the 1990s, some support for graphics and color was added, but still following the same paradigm and aimed at printing on physical media, now including overhead projector slides.

The move beyond paper came soon after this, with hyperlinks and other web publishing support getting layered on top of basic LaTeX; but these (together with many other subsequent extensions) were never incorporated as integral parts of its design. This piecemeal development has resulted in the extensions being very fragile so that they often break in more complex documents, especially when used in combination.

Even today it is the case that, in many areas, core LaTeX provides little in the way of APIs that are robust enough to be safely used or built on by developers. Therefore most such extensions are forced to contain a lot of code that overwrites many internal LaTeX commands: such unfortunate and unsustainable implementation practices being currently necessary in order to implement most extra functionality. Such methodology naturally leads to many issues, of which a common and particularly frustrating one is incompatibilities between extensions.

Another severe limitation of current LaTeX is that, whenever it produces an output page, it very carefully throws away the wealth of structural information it has used and accumulated while producing that page.[1] As a result, a PDF or DVI file produced by basic LaTeX is nothing but a stream of positioned glyphs containing very little structural information.

As long as your intention is only to print a document on a physical medium, then this is all that is required. However, for quite a while now other uses of documents have been increasing in importance so that nowadays many documents are never printed, or printed only as a secondary consideration.

Coming into the 21st century, for many reasons great interest has arisen in the production of PDF documents that are "accessible", in the sense that they contain information to assist screen reading software, etc., and, more formally, that they adhere to the PDF/UA (Universal Accessibility) standard [17], explained further in [3]. Ross Moore has recently written a comprehensive introduction to this subject [9].

[1] This jettisoning of much useful information was absolutely essential during the early decades of LaTeX development because the system would otherwise never have run in the severely limited memory available on the then current computer systems. So all information no longer needed for producing a "printed page" was dropped immediately so as to keep the memory footprint within reasonable limits.

But even that was not always enough: on his first PC one author was greeted with "out of memory" while loading the `article` class.

One important requirement for such PDF documents is that they must contain a significant amount of embedded structural information and other data. At the moment, all methods of producing such "accessible PDFs", including the use of LaTeX, require extensive manual labor in preparing the source or in post-processing 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. Such methods and their inherent problems have been discussed and demonstrated recently by developers such as Ross Moore, much of whose work in this area can be accessed through the web site at [10].

The LaTeX Project Team have for some years been well aware that these new usages are not adequately supported by the current system architecture of LaTeX, and that major work in this area is therefore urgently needed to ensure that LaTeX remains an important and relevant document source format. However, the amount of work required to make such major changes to the LaTeX system architecture will be enormous, and definitely way beyond the regular resources of our small team of volunteers working in their spare time! Or maybe just about possible, but only given a very long — and most likely too long — period of time.

At the TUG conference 2019 in Palo Alto our previously pessimistic outlook on this subject became cautiously optimistic, because we were approached by two senior engineers from Adobe to discuss the possibility of producing structured PDF from LaTeX source without the need for the normal requirement of considerable manual post-processing. Moreover, they had come to ask us what it would take to make this happen. They indicated that, if this turned out to be a feasible project for which we could provide them with a convincing project plan, then they would happily advise Adobe, at a suitably influential level of management, to support such work, both financially and in other ways — no strings attached.[2]

As a result of these discussions, towards the end of 2019 the authors, together with Ulrike Fischer, produced an extended feasibility study for the project, aimed primarily at Adobe engineers and decision makers. This study [8] describes in some detail the various tasks that will constitute the project and their inter-dependencies. It also contains a plan covering how, and in what order, these tasks should be tackled in order both to achieve the final goal and, at the same time, to provide intermediate concrete results that are relevant to user communities (both LaTeX and PDF); these latter will help in obtaining feedback that is essential to the successful completion of later tasks.

The project plan gained the approval of the Adobe management and we agreed in early 2020 to start the project with Adobe contributing a substantial portion of the expected project costs. But this was 2020, so then "Corona arrived" with its huge spanner to throw into the works! Thus all activity came to a halt while Adobe was forced to reassess such external financial commitments. However, we were quite quickly told that this was a temporary setback, so we restarted work on the project, albeit at a slightly slower pace, and so were able to present our first results at the TUG 2020 online conference: i.e., the new hook management system for LaTeX.[3]

2 What the project covers

This project has the title "LaTeX Tagged PDF" so before discussing further its coverage we should describe the use of the word "Tagged" here. A good starting point for this is the following quote from the PDF standard [2, 15] which defines tagging in the PDF context thus:

> Tagged PDF ([introduced in]PDF 1.4) is a stylized use of PDF that builds on the logical structure framework described in Section [..], "Logical Structure." It defines a set of standard structure types and attributes that allow page content (text, graphics, and images) to be extracted and reused for other purposes. It is intended for use by tools that perform the following types of operations:
>
> - Simple extraction of text and graphics for pasting into other applications
> - Automatic reflow of text and associated graphics to fit a page

[2] Adobe's rationale for this support is that LaTeX is currently a very important document format in the STEM disciplines and is one of the formats best suited to produce well-structured PDF "out of the box", given that its source already contains such a large amount of information about the document's structure. Thus, with LaTeX capable of supporting the straightforward production of structured PDF documents, a large number of such documents would become available on the Internet. In addition, many older documents could be reasonably easily reprocessed to add this structural information to their PDF form. For Adobe all this is of great interest as the existence of large numbers of such documents forms an important and strategic input to their Document Cloud [1] offerings and related future services.

[3] The project task for this is described in section 2.2.5 of the Feasibility Study [8].

> of a different size than was assumed for the original layout
> - Processing text for such purposes as searching, indexing, and spell-checking
> - Conversion to other common file formats (such as HTML, XML, and RTF) with document structure and basic styling information preserved
> - Making content accessible to users with visual impairments (see Section […], "Accessibility Support")

Based on this definition we can now give a high-level description of this project by analyzing the quoted text from the perspective of how LaTeX currently works and how we will need to change it. The quoted text tells us that to produce a Tagged PDF document from a LaTeX source we must proceed as follows:

First, we must exploit LaTeX's knowledge of how the components of a document are presented in the LaTeX source file and used to describe its structure, consisting of "structure types and attributes" such as the paragraph text, headings, lists, items, graphics, tables, table cells, mathematics, etc.

Then, we must use this knowledge about the source document to add the necessary components to the PDF we are constructing. There are two basic ingredients of this: a tree structure whose nodes represent the "logical components" of the document; and a mechanism for identifying those portions of the "page content" that correspond to each (leaf) node in this tree.

These "tagged" portions of the content, together with information about the corresponding structure node, can then be used by other (consumer) applications to carry out operations like those listed at the end of the quote above.

None of the above may sound too complicated given that LaTeX knows what kind of structures will be in the source file that it is processing. But as we explained earlier, for performance reasons (necessary when LaTeX was designed and implemented), currently this structural information is disposed of as soon as possible — therefore, by the time of writing out the content of a page to the PDF file it is no longer available.

In addition, there are some problematic details within these processes. For example, the main content of a PDF document is a collection of "page objects" and the "tagged portions" of the text have to be nested within each page object; but TeX's asynchronous pagination process does not play well with this constraint. Dealing with this problem is therefore part of one of the lower-level tasks in the project, one that may best be solved by enhancements at the engine level.

For this reason, a large portion of the necessary project work consists of adding to the core of LaTeX all the functionality, data structures and interfaces that are necessary for preserving and processing the many details of this structural information. This will result in a much enhanced LaTeX system that can manage and transport structural information from a source to an output format.

With all of this firmly embedded into LaTeX, the project will be able to go further to provide, for example, appropriate document-level interfaces for some types of data that currently are not, at least by default, made explicit in the source file. In this area there already exist some solutions for aspects such as metadata, PDF bookmarks, etc., produced by Ross Moore, Scott Pakin, Heiko Oberdiek and others [11, 12, 14]. Wherever feasible, we shall unify and incorporate such useful existing work. Support for some, but not all, of the wider (beyond tagging) requirements of "Accessible PDF" will also be included in the project work.

We also want to include here some important (to us, at least) observations concerning the processing of structural information using LaTeX. From the document-level (frontend) processing to the internal data structures and interfaces, everything we wrote in the high-level description above is essentially independent of the target output format (Tagged PDF in this project). Identical processing will be needed for any output format that needs such structural information, including HTML, XML and other "tagged/structured formats".

Thus, while the project will focus primarily on PDF output (either generated directly by the TeX engine or through a DVI-based workflow) it will, as a bonus, make it easy to add other such output formats to the workflow by simply replacing an output (backend) module. Instead of PDF output, HTML5 or some other format will then get written. As of now, such alternative backends are not part of the project coverage, but once LaTeX is able to pass structure information with well-defined interfaces to a backend we expect that support for other structured output formats will follow. Such work may be undertaken by us or by other teams, possibly in parallel to the later phases of the project discussed in this article.

3 Explaining the blueprint

The document entitled "LaTeX Tagged PDF Feasibility Evaluation" [8], available from the LaTeX Project website [5], is a forty-page study that explains in detail both the project goals and the tasks that need to be undertaken, concluding with a plan for how we think the project should be undertaken. Thus it is divided into three parts, beginning with an "Introduction" which contains an 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 PDF.

It is advisable, if you are consulting this study, to bear in mind that it was addressed primarily to an audience within Adobe;[4] this consisted of engineers and managers with a wide knowledge of digital typography and electronic publishing but not necessarily much background within the specialized world of TeX, LaTeX and friends.

3.1 The major tasks

Following the overview, the next part of the study ("Project Overview") documents all the major tasks that we consider necessary to reach the project goals. These tasks are categorized under five headings, the two most important being "General LaTeX Extension Tasks" and "Structured PDF Tasks".

For each task, the study describes the rationale for its inclusion and the work that has to be undertaken. For the larger tasks, suitable subtasks are also identified. Furthermore, to support the project timeline planning, all the dependencies of each task on other tasks are documented. Each such task section ends with a list of deliverables, which will help us to measure, and hence monitor, our progress through the project.

Note that in total there are twenty engineering tasks for which we believe that the necessary work is well understood and therefore it "only" has to be done — BUT done very well! In addition there are a few project tasks that will require more extensive research. Their descriptions in the study are much less complete; this is simply because the engineering details are not as yet known, or not well enough understood. These research tasks are, naturally, likely to identify further engineering tasks.

Note that, since the tasks in this part are arranged by category, they are not listed in their chronological order of execution within the overall plan: i.e., a task's number is no indication of how far along in the project that task will be tackled.

[4] The public, cited, version of the study was updated in September 2020 with some minor redactions, corrections and clarifications.

3.2 The project plan

In the final major part of the study ("Project Timeline") we develop a complete project plan consisting of six phases. The tasks have been sorted into these phases in such way that the dependencies between them are respected; but also, and more importantly, we have ensured that the results of each phase will offer immediate benefits to the LaTeX community.

This latter requirement is important since it will lead to timely feedback and early adoption. As you can see below, we expect, for example, that on completion of phase II it will already be possible to automatically generate tagged PDFs for a restricted set of documents. In later phases this level of automation will be extended to a wider range of documents.

Phase I — Prepare the ground

This phase covers some tasks that are important precursors of later work; it is already well under way. One important deliverable here is the new general hook management system for LaTeX that was presented at the 2020 TUG conference and which is now part of core LaTeX, starting from the October 2020 release.

Another task here is the initial work on the low-level requirements for the creation of tagged PDF; this is currently available in the highly experimental package `tagpdf`, which uses prototype code from `pdfresources`-related files for the creation and management of PDF objects. This package thus supports such tasks as the creation in the PDF of the structure tree and adding the necessary connections between the nodes in this tree and the "content stream" of each page.

It currently enables the creation (with some user intervention) of tagged PDF documents; *but*, please note that this work is truly experimental and so both the code and interfaces are likely to change, or even vanish, at any time.

This phase is the only one that was not designed to offer significant benefits to users/authors. Nevertheless, the standardization and interfaces provided by the general hook management will arguably benefit these groups in addition to its direct target, the package developers.

Phase II — Provide tagging of simple documents

The main goal of phase II is to provide basic support for the automation of the tagging process. In this phase the automation will cover only relatively simple documents: those that do not contain any of the more complicated structures such as mathematics and tables.

At this stage, the automated tagging will be provided by loading an add-on package rather than its being in the kernel, and it will be only a prototype implementation.

For this phase several "workarounds" will be required in order to provide necessary, but missing, features. For example, instead of setting up (as part of this phase) a full extended cross-reference system (similar to the `zref` package by Heiko Oberdiek, but built into core LaTeX) we will either use that package or add temporary code written to "work around" any issues resulting from this functionality not yet being a component of the kernel. Any such "workaround code" will need to be removed or adjusted in a later phase.

This ordering of the work was chosen so as to offer tangible results reasonably early in the life of the project, rather than taking up too much time at this stage on making purely internal improvements that have no immediately visible application and therefore appear to many users to be devoid of value.

Phase III — Remove the workarounds needed for tagging

The main goal of phase III is to extend the coverage of automatic tagging to a wider variety of documents by making further basic document elements "tagging aware". These extensions will also enable us to remove the "workarounds" introduced (to provide a working prototype) in the previous phase. This phase will also include a metadata mechanism that is integrated into the LaTeX kernel.

Phase IV — Make basic tagging and hyperlinking available

The main goal of phase IV is to incorporate into the kernel all the code from the prototype packages. This needs to be done very carefully and cautiously as there should be no negative impact on the processing of legacy documents. Thus we expect to need at least one full LaTeX release cycle[5] to complete this work.

Phase V — Extend the tagging capabilities

With basic tagging available, the focus of phase V is to provide extended support for tagging by adding tables and formulas (mathematics) to the supported document structures.

At this stage, interfaces will also be added wherever needed to support other aspects of Tagged PDF such as the specification of alternate text (for formulas, illustrations, etc.).

[5] See Section 3.3 for further information concerning phases and this release cycle.

Phase VI — Handle standards

Finally, phase VI focuses on the provision of support for the relevant PDF standards (such as PDF/A [16], explained further in [13]; and PDF/UA [17], see also [3]) at least so far as this is possible using LaTeX directly, i.e., without any post-processing of the PDF file. It also covers kernel support for some further features such as the production of outlines for a document and the incorporation into a PDF of "associated files".

Research work

In addition to these six phases, which contain only tasks that are largely understood from a technical perspective, there are a number of other tasks that will require preparatory research in order to understand what engineering work is needed. This research will be carried out in parallel with the other work and, as indicated earlier, it is very likely to lead to the specification of additional engineering tasks. Thus the research outcomes will probably lead to some alterations and extensions in phases IV to VI.

Alterations, adjustments and reporting

On the whole we believe that this plan offers a consistent and sensible approach to attaining the intended goals of the project. However, there will undoubtedly be some changes since, for example, the research tasks will probably lead to some augmentation of the planned tasks and changes to some of them. Moreover, we have already found that we want to move some deliverables (from the larger tasks) forward to earlier phases. Also, in some cases we are thinking about not completely finishing a task within the planned phase because it appears more important to first focus on other work (of course, such changes will work only in cases where those deliverables are not required for other tasks). Furthermore, working through a task may identify some additional work that is needed but not yet accounted for, leading to new subtasks with extra deliverables.

We are therefore planning to provide an appropriate level of continuing (public) reports on the project's status and results, including any additions or changes of direction. The Feasibility Study, with its fairly detailed documentation and deliverables, will provide a good starting point for this important activity. The exact form and shape of this reporting is not yet decided, but we will in due time announce it on the LaTeX Team website [5] and/or in *TUGboat* as appropriate.

3.3 Timeline and other considerations

The time estimates we supplied (in the Feasibility Study) for the individual tasks are based on the assumption that there will be *at least* (the equivalent of) one software developer working exclusively and full time on a task throughout its allocated time period, together with additional support from further developers as necessary. We are also very well aware that activities such as documentation and testing require substantial specialized professional input that will also need to be resourced.

Whether or not this level of effort will be possible at all times will depend on various factors, an important one being the availability of appropriate funding that can free the responsible developers and other key workers from the need to undertake other work to earn a living. While the financial sponsorship from Adobe will go a long way towards meeting the needs of the project, it may well not be sufficient by itself to fully sustain the work. For all the above reasons, the estimates in the study should be considered as approximations and not as definite values.

A realistic scenario would be that each phase takes between one and two release cycles of LaTeX, of which there are two per year. This implies that the project will stretch across four years as a minimum, but it most probably will be somewhat longer. Additional funding will help to ensure timely delivery of each phase and may additionally allow us to broaden the scope in some areas. Nevertheless, given the complexity of the topic, any expectation of earlier delivery dates is not realistic.

It is also important to note that all the necessary updates to important external packages (those not supported by the LaTeX Project Team) are expected to be done using external resources, i.e., by the maintainers of those packages. This assumption is probably not tenable in all cases (see, for example, the discussion in [8, Task 2.4.3]). In such cases, additional work will have to be undertaken as part of the project, which will also alter the timeline or require hiring additional developers to take on such work.

Last but not least, the success of the project will also depend very much on productive collaborations with many people in the LaTeX community: testers, package writers, and possibly also those who tend to the underlying TeX engines and the various utility programs used to produce PDF output. Also, from the wider world, we shall need input from a variety of experts in the production of high-quality PDF and from those with knowledge of how consumer applications use its features in the real world.

4 First results

As mentioned earlier, we have started the project despite COVID-19 getting in the way. So we can now already report on some success stories:

- The new LaTeX hook management system (Task 2.2.5 of [8]) presented at TUG 2020 [7] and in a *TUGboat* article [6];
- PDF string support (Task 2.2.1 of [8]) which is largely an internal enabler for later tasks;
- Initial experiments with `tagpdf` prototype code (part of Task 2.2.6 of [8]). See [4] for a discussion.

5 Outlook

Right now we are in the middle of phase I and expecting it to be finished with the Spring release of LaTeX in 2021. Given that we intend to shift the LaTeX release dates next year to better align with the yearly TeX Live distributions, we expect to start work on some tasks of phase II already before the next LaTeX release.

In addition to the ongoing work on engineering tasks, some effort will go into managerial tasks, e.g.,

- Setting up connections and collaboration with external experts to gain expertise in areas where the present team is lacking, and to ensure that our work will continue to be focused on pertinent goals;
- Looking for additional financial support to bring in extra expertise and hence, we hope, speed up the later phases;
- Setting up a professional system for the production of documentation, high in both quality and quantity.

Of course, LaTeX development and maintenance will not be solely restricted to this project over the coming years. There are many other activities that the LaTeX Project Team plans to carry out in parallel. Their results will, as in the past, appear via the usual communication channels, e.g., our website, `ltnews` newsletters, *TUGboat* articles and/or Internet-based discussions on StackExchange chat, `LATEX-L` and other places.

References

[1] Adobe Document Cloud. `https://en.wikipedia.org/wiki/Adobe_Document_Cloud`.

[2] Adobe Systems Inc. *PDF Reference 1.7*, Nov. 2006. `https://www.adobe.com/devnet/pdf/pdf_reference.html`. Freely available version of [15].

[3] O. Drümmer, B. Chang. *PDF/UA in a Nutshell — Accessible documents with PDF*. PDF Association, Aug. 2013. https://pdfa.org/resource/pdfua-in-a-nutshell/.

[4] U. Fischer. Creating accessible pdfs with LaTeX. *TUGboat* 41(1):26–28, 2020. https://tug.org/TUGboat/tb41-1/tb127fischer-accessible.pdf

[5] LaTeX Project Team. Website of the LaTeX Project. https://latex-project.org/.

[6] F. Mittelbach. Quo vadis LaTeX(3) Team — A look back and at the upcoming years. *TUGboat* 41(2):201–207, 2020. https://latex-project.org/publications/indexbyyear/2020/.

[7] F. Mittelbach. Video: Quo vadis LaTeX(3) Team — A look back and at the upcoming years. 2020. Recording of the talk given at the online TUG 2020 conference, https://youtu.be/zNci4lcb8Vo.

[8] F. Mittelbach, U. Fischer, C. Rowley. *LaTeX Tagged PDF Feasibility Evaluation*. LaTeX Project, Sept. 2020. https://latex-project.org/publications/indexbyyear/2020/.

[9] R. Moore. Implementing PDF standards for mathematical publishing. *TUGboat* 39(2):131–135, 2018. https://tug.org/TUGboat/tb39-2/tb122moore-pdf.pdf

[10] R. Moore. Website: Tagged PDF examples and resources, 2020. http://maths.mq.edu.au/~ross/TaggedPDF and in particular /TUG2019-movies.

[11] R. Moore, C.V. Radhakrishnan, et al. Generation of PDF/X- and PDF/A-compliant PDFs with pdfTeX — pdfx.sty. https://ctan.org/pkg/pdfx, Mar. 2019.

[12] H. Oberdiek. The bookmark package. https://ctan.org/pkg/bookmark, Dec. 2019.

[13] A. Oettler. *PDF/A in a Nutshell — PDF for long-term archiving*. PDF Association, May 2013. https://pdfa.org/resource/pdfa-in-a-nutshell/.

[14] S. Pakin. The hyperxmp package. https://ctan.org/pkg/hyperxmp, Oct. 2020.

[15] Technical Committee ISO/TC 171/SC 2. *ISO 32000-1:2008 Document management — Portable document format (PDF 1.7)*, July 2008. https://iso.org/standard/51502.html. Freely available as [2].

[16] Technical Committee ISO/TC 171/SC 2. *ISO 19005-2:2011 Document management — Document file format for long-term preservation — Part 2: Use of ISO 32000-1 (PDF/A-2)*, July 2011. https://iso.org/standard/50655.html.

[17] Technical Committee ISO/TC 171/SC 2. *ISO 14289-1:2014 Document management applications — Electronic document file format enhancement for accessibility — 1: Use of ISO 32000-1 (PDF/UA-1)*, Dec. 2014. https://iso.org/standard/64599.html.

◇ Frank Mittelbach
Mainz, Germany
frank.mittelbach (at) latex-project dot org
https://www.latex-project.org

◇ Chris Rowley
Sattahip, Chonburi, Thailand
chris.rowley (at) latex-project dot org
https://www.latex-project.org