LaTeX News

Issue 40, November 2024 (LaTeX release 2024-11-01)

Contents

Thirty years of LaTeX 2ε

In summer 1994, i.e., thirty years ago, LaTeX 2ε saw its first public release. Back then it was meant to be an intermediate version (hence the ε) on the way to a major new version (the mythical LaTeX3) that we expected to take a couple of more years to reach maturity. It took much more than that in the end—nominally, LaTeX 2ε is still with us today.

However, under the hood, LaTeX 2ε changed a lot throughout these thirty years, as one can see, for example, when looking through the forty newsletters [2] that accompanied the LaTeX releases that happened in the meantime.

During the first two decades, the LaTeX kernel was kept largely stable with only minimal bug fix activities. During that period additional functionality was mostly provided through new or extended packages that could be loaded in the document preamble. This included many of the ideas targeted for LaTeX3, e.g., expl3 (LaTeX3 programming language), xparse (new document command interface), xtemplate (a configuration mechanism), and many others.

Initially, this approach worked well and provided good backward compatibility; however, over time it became apparent that keeping all developments confined to packages was more and more problematical. Features or bug fixes that should have been generally available, i.e., part of the kernel, were only available in packages, so a lot of dependencies between packages were introduced and resulted in convoluted code that was difficult to manage. For example, hyperref had to rewrite a lot of kernel (and package) macros, so the code and behavior of other packages had to change depending on whether or not hyperref was loaded or not.

Thus, in 2015 the LaTeX team decided to change the policy and (re)start active kernel development, see [3]. To ensure continuous backward compatibility we introduced at the same time the latexrelease package that enables users to roll back changes to the LaTeX kernel to an earlier release, in case this is necessary to successfully rerun a document produced at that time.

As a consequence of this policy change the last decade saw a larger number of enhancements and corrections that were made part of the LaTeX kernel. Overall, we can confidently say that the new approach has worked well and enabled us to modernize LaTeX and ensure that it remains relevant without compromising one of

the cornerstones of LaTeX: its outstanding ability to reprocess old documents written many years ago.

Being able to update and modernize the kernel sources allowed us to embark in 2019 on the multi-year "LaTeX Tagged PDF" project with the goal of automatically providing accessible PDF documents with LaTeX. While there are several more project phases to complete, the milestones already reached allow users to generate PDF/UA compliant documents if the input is restricted to a (growing) subset of packages and document classes; see next section and previous newsletters.

A big change happened with the 2020-02-02 release as part of the project activity, albeit somewhat obfuscated by us as "Improved load-times for expl3". While technically correct, what it really meant is that we had finally integrated the programming layer of LaTeX3, i.e., the ideas originally sketched out around 1992. Or saying it differently: with that date the original ideas for LaTeX3 became a reality as part of the standard LaTeX kernel.

With the programming layer available under the hood we were then able to provide new concepts and extensions as part of LaTeX, e.g., the hook management system, a new mark mechanism, core functionality for tagging and PDF resource management, a consistent key/value interface, and more recently the socket and plug mechanism.

More will follow while we continue to work on modernizing LaTeX and bringing the Tagged PDF project to a truly successful completion—so stay tuned and watch this space for future announcements in the next newsletters.

News from the "LaTeX Tagged PDF" project

Engine support: An important update

As detailed below, work is progressing on the Tagged PDF project. There are many drivers for this work, including legal changes in many places which will increasingly require well-tagged PDFs including full support for mathematics. As part of the work on this, we are looking at the technical abilities of the TeX engines.

With XeTeX, it is impossible to reliably produce tagged PDFs due to engine limitations. The increasing importance of tagged PDFs means that this requires a move away from XeTeX. We will continue to address issues with XeTeX support in team-maintained LaTeX code on a best-effort basis. No *new* functionality will be added for XeTeX by the LaTeX team. It is likely that over time functionality may become more restricted, and users are urged to migrate XeTeX documents to LuaTeX.

For pdfTeX, tagging is available and we are able to support mathematics by including relevant TeX source or by using externally-generated MathML. Only LuaTeX is capable of *automatic* generation of MathML as part of a LaTeX run. Thus pdfTeX continues to be supported for existing material, but for new documents, moving to LuaTeX is recommended.

We cannot make statements about the support for other engines such as (u)pTeX, as we don't use these programs nor have in depth knowledge of their functionalities. To the best of our knowledge, core LaTeX works well with these engines, but if and to what extent tagging can be supported will remain to be seen. If relevant information becomes available to us we will provide an update in future editions of the LaTeX newsletter.

Tagging support for external packages

At `https://latex3.github.io/tagging-project/tagging-status/` we show the status of many LaTeX packages and classes with respect to PDF tagging. We also started to improve tagging support in external packages. If the `firstaid` key is used in addition to the `phase-III` key, basic commands of several packages, including amsthm and fancyvrb, can now be used.

Improved table tagging

The tagging of tabulars has been extended: it is now possible to tag row headers and to create cells that span more than one row.

The interface to this functionality is not finalized but can be accessed in the current release by specifying the row and columns to be treated as headers. For example

```
\tagpdfsetup{
 table/header-rows={1,2},
 table/header-columns={1} }
```

would specify that in the following tables the first two rows and first column of each row should be tagged as heading entries.

Similarly you may add a RowSpan attribute to tag a cell that spans two rows using:

```
\tagpdfsetup{table/multirow={2}}
```

Automatic MathML tagging

When LuaLaTeX is being used, and the luamml package is available, and if the document uses the unicode-math package, then the math module will automatically convert each math formula to MathML and use it to attach MathML associated files (or MathML Structure elements) to the tagged PDF. This new feature can be disabled with `\tagpdfsetup{math/mathml/luamml/load=false}`. More options to configure MathML tagging can be found in the documentation of latex-lab-math.

Change behavior of tagging sockets with two arguments
When calling tagging sockets with two arguments using `\UseTaggingSocket` when tagging is suspended, previous versions of LaTeX 2ε dropped both arguments. This behavior has been changed to drop the first argument and preserve the second one instead, thereby allowing tagging sockets to be used to wrap existing content which should still appear in a non-tagging context.

Since no tagging sockets currently provided by LaTeX use two arguments we do not expect this change to affect any existing documents, but if a custom tagging socket has been defined outside of the kernel it might need to be adapted to be compatible with the new behavior.

(github issue 1500)

Changes to the LaTeX kernel

Handling paragraph continuation
Already LaTeX 2.09 offered some automation to detect whether or not text after a list or some other display environment is meant to be a continuation of the current paragraph or should start a new one. The document-level syntax for this is that a blank line after such an environment signals to LaTeX that it should start a new paragraph; whilst no blank line signals that there should be no new paragraph and the text should be considered a continuation.

Unfortunately, there were a number of cases where the original 2.09 approach failed, e.g., with

```
{⟨local customizations⟩
\begin{equation} a<b \end{equation}}
⟨some text⟩
```

the ⟨*some text*⟩ incorrectly started a new paragraph. Bug reports about this behavior can be traced back to the time LaTeX 2ε was developed, e.g., one test file from 1992 has a note that the above case was unfortunately not resolvable despite some improvements made back then. The main cause of the issue (as you probably guessed) is that the mechanism failed whenever the environment was executed within a group (`{...}`, `\begingroup`/`\endgroup`, or `\bgroup`/`\egroup` pair) that was closed before the next blank line was reached.

While most of the time this could be visually corrected by adding some explicit `\noindent`, the situation got worse when we tried to implement tagged PDFs resulting in incorrect structures or worse.

We therefore made a new attempt to resolve this problem in every situation and this new solution is rolled out in the current release.

Avoid bogus "no item" error
The commands `\addvspace` and `\addpenalty` generated the famous error message "Something's wrong—perhaps a missing `\item`" when they were encountered outside vertical mode. Most of the time this error was bogus and if not, then it was generated several times rather than once.

Once upon a time (in LaTeX 2.09) it was necessary that these commands were used only in vertical mode, but with LaTeX 2ε in 1994, we changed the internals but simply overlooked that this error message then had become useless. In this release, i.e., 30 years too late, we have finally lifted the ban and from now on this error should only show up if there is indeed a missing `\item`.

(github issue 1460)

Switch to T1 as default encoding in documents using `\DocumentMetadata`
As it is well known, the font encoding `OT1` supports only 128 characters and has various problems and quirks notably for languages different to English. Nevertheless `OT1` is the default encoding in LaTeX and this cannot be easily changed without affecting many documents as the `T1` version of the fonts have slightly different metrics.

The introduction of the `\DocumentMetadata` command, which announces *new* code and changes that can also affect the layout gives us now the opportunity to make this step. So with this version a use of `\DocumentMetadata` with (pdf)LaTeX will setup `T1` as default font encoding.[1] To ensure that scalable fonts are used, the package `cm-super` has to be installed. Users who want to revert to the `OT1` encoding in their document can do so with `\usepackage[OT1]{fontenc}`.

Code improvements

Avoiding key–value option clashes between classes and packages
In LaTeX News 35 [5] we introduced key–value option processing to the kernel. Following the standard for LaTeX 2ε options, keyval options given to the `\documentclass` line were treated as global and so parsed by every package. However, with keyvals, the likelihood of a name clash between a class-specific option and one used by a package is much higher than it is with simple strings. We have therefore refined the mechanism in the current release.

When a class uses the kernel keyval processor, any options it recognizes are recorded and any packages using the keyval processor will then *skip* these "global" options. To allow for the case where a class directly uses an option which should be global (for example `draft`), a new key property `.pass-to-packages` has been added. This can then be set to indicate that this key is not to be skipped. For example

```
\DeclareKeys{
  draft .if = {ifl@cls@draft},
  draft .pass-to-packages = true,
```

[1] The Unicode engines will continue to use `TU` as the encoding.

```
  mode  .store = \cls@mode
}
```

in a class would create two options, `draft` and `mode`. The `draft` option will be treated in the normal way by packages using keyvals, but they will ignore the `mode` option: it is effectively marked as "private" to the class. *(github issue 1279)*

Improvement to XeTeX `\showhyphens`

When using `\showhyphens` with XeTeX, missing character warnings would be generated for any character not in Latin Modern. This has been corrected and the warnings are suppressed. *(github issue 1380)*

Improved error raised for empty hook name

When using the hook management, both hook and label names (if specified) should be non-empty. Before, empty hook and empty label names both raised the same label-specific error:

```
! LaTeX hooks Error: Empty code label on line ..
  Using 'top-level' instead.
```

This has now been improved. Now an empty hook name generates

```
! LaTeX hooks Error: Empty hook name on line ..
```

(github issue 1423)

Provide counter representations for link targets

To create unique target names for links the package hyperref uses a special counter representation `\theH`⟨*counter*⟩. To ensure that this counter representation exists, hyperref redefined the commands `\@definecounter`, `\@addtoreset` and `\refstepcounter`. This counter representation is also needed for the Tagged PDF project and so these augmented command definitions have now been incorporated into the kernel. Thus from now on every `\newcounter{`⟨*counter*⟩`}` will define not only `\the`⟨*counter*⟩ but also `\theH`⟨*counter*⟩.

Extending `\refstepcounter`

For many years, the package hyperref had been redefining `\refstepcounter` and adding code that creates link targets. The kernel definition has now been extended with socket interfaces that will allow hyperref to avoid the redefinitions. The new interfaces are also used by the Tagged PDF code that needs target names to resolve references between structures.

Bug fixes

Fix wrong file type in a rollback warning

When LaTeX is rolled back to date ⟨*date1*⟩ and a class or package with minimum date requirement ⟨*date2*⟩ is to be loaded, a rollback warning is raised if ⟨*date2*⟩ is later than ⟨*date1*⟩:

```
LaTeX Warning: Suspicious rollback/min-date
               date given.
  A minimal date of YYYY-MM-DD has been
  specified for package '<pkgname>'.
  But this is in conflict with a rollback
  request to YYYY-MM-DD.
```

In some cases this message showed a wrong file type, i.e., `document class '<pkgname>'` or `package '<clsname>'`. This has now been corrected. *(github issue 870)*

Fix existence check of document environments

`\NewDocumentEnvironment` and friends define (or redefine) a document environment using the space-trimmed ⟨*envname*⟩, but the existence check for ⟨*envname*⟩ was done without space trimming. Thus when the user-specified ⟨*envname*⟩ consists of leading and/or trailing space(s), it may lead to erroneously silent environment declaration. For example, in

```
\NewDocumentEnvironment{myenv}{}{begin}{end}
\NewDocumentEnvironment{␣myenv␣}{}{begin}{end}
```

the first line defines a new environment `myenv` but the second line would check existence for `␣myenv␣` (which is not yet defined), then redefine `myenv` environment without raising any errors. This has now been corrected. *(github issue 1399)*

Handling of global keys with spaces

If the global (class) options contained spaces around key names, `\ProcessKeyOptions` would fail to remove known keys from the list of unused global options and `\OptionNotUsed` would mistakenly add space-surrounded key names to that list. The first issue was corrected as a hotfix in patch level 1 of the November 2023 release (but unfortunately not mentioned in [6]) and the second in the current release. *(github issue 1238)*

File list entries for rolled back packages/classes

When the rollback mechanism for packages and classes was introduced in 2018 [4], loading of the selected historic release was not recorded in the file list used by `\listfiles`. This has now been corrected so that the extended usage [7]

```
\listfiles[hashes,sizes]
```

now gives more complete and less confusing info. *(github issue 1413)*

doc: `\PrintDescribeMacro` in preamble

In doc version 2 it was possible to alter the definition of `\PrintDescribeMacro` and similar commands in preamble. In version 3 this stopped working because they were reset at the end of the preamble. This has now been implemented differently and changes in the preamble are possible again. *(github issue 1000)*

Avoid low-level error if \ShowHooks is used late
If \ShowHooks was used to examine a package hook after the package was loaded, a low level error resulted. This has now been corrected. *(github issue 1513)*

Avoid code duplication in rollback
When the kernel uses \AddToHook in a region that might be rolled back (which happens in a few places) and a document requests a rollback, then we have the situation that the hook already contains code to which we added the same (or slightly different) code during the rollback; this results in code duplication or, worse, in errors. This has now been corrected by dropping any such code chunk (if there is one) prior to adding the rollback code. *(github issue 1407)*

Passing template keys using \KeyValue
With the move of the template code to the kernel, internal functions were reviewed to improve efficiency. However, there was an oversight in how passing key values from one setting to another was implemented, such that using \KeyValue could result in an infinite loop. This has now been fixed. *(github issue 1486)*

Changes to packages in the amsmath category

Extend support for \dots
The implementation of \dots in amsmath has the feature that it selects different dots depending on the symbol that follows: e.g., dots between commas would normally be on the baseline, while dots between binary or relational symbols would be raised. However, when symbols such as \cong were protected from expansion in moving arguments (so that they worked in places such as headings) it had the unfortunate side-effect that the \dots magic stopped working for them. This has now been corrected. *(github issue 1265)*

Changes to packages in the tools category

Modification to generation of the .tex from fileerr
The fileerr extraction has been modified to write rename-to-empty-base.tex rather than .tex to comply with an expected security change in TeX Live 2025. The build.lua file for the tools has been modified to rename rename-to-empty-base.tex to .tex after unpacking. However if using docstrip directly rather than using l3build or the unpacked zip file from CTAN, the user must now rename the file and install as .tex. *(github issue 1412)*

array: Improve >{...} specifier
If the argument of >{...} ended with a command accepting a trailing optional argument, e.g., defined for example with \NewDocumentCommand\foo{o}{...}, one could get low-level parsing errors. This has now been corrected. *(github issue 1468)*

array: Tagging support for \cline
In the last release we added tagging support for array, longtable and other tabular packages, but we overlooked that the kernel definition for \cline also needs modification because the rule generated by the command needs to be tagged as an artifact. Furthermore, the processing of a \cline looks to the algorithm as if another row is added (which is technically what happens), thus it was also necessary to decrement the internal row counter to get a correct row count. This has now been corrected in array, which is automatically loaded for tagging, so that all these packages are now fully compatible with the tagging code if it is turned on. *(github tagging issue 134)*

longtable: Extend caption type
The longtable package has been extended and now provides the command \LTcaptype (stemming from the ltcaption package) to change the counter and caption type used by the \caption command from longtable. So with \renewcommand\LTcaptype{figure}, a longtable will step the figure counter instead of the table counter and produce an entry in the list of figures. An empty definition, \renewcommand\LTcaptype{}, will suppress increasing of the counter. This makes it easy to define an unnumbered variant of longtable:

```
\newenvironment{longtable*}
  {\renewcommand\LTcaptype{}\longtable}
  {\endlongtable}
```

longtable: Prevent \pagegoal exceeding maximum value
An internal guard has been added to avoid TeX errors if \pagegoal is increased beyond the maximum value for a TeX dimension. *(github issue 1495)*

Changes to l3build

To support third-party developers testing their code against pre-release LaTeX, a new switch --dev has been added to l3build. This allows the developer to run

```
l3build check
```

to run their test suite against the current release of LaTeX and

```
l3build check --dev
```

to run exactly the same tests using the development release of LaTeX.

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

[2] LaTeX Project Team. *LaTeX 2ε news 1–39*. June 2024. https://latex-project.org/news/latex2e-news/ltnews.pdf

[3] LaTeX Project Team. *LaTeX 2ε news 22*. January 2015. https://latex-project.org/news/latex2e-news/ltnews22.pdf

[4] LaTeX Project Team. *LaTeX 2ε news 28*. April 2018. https://latex-project.org/news/latex2e-news/ltnews28.pdf

[5] LaTeX Project Team. *LaTeX 2ε news 35*. June 2022. https://latex-project.org/news/latex2e-news/ltnews35.pdf

[6] LaTeX Project Team. *LaTeX 2ε news 38*. November 2023. https://latex-project.org/news/latex2e-news/ltnews38.pdf

[7] LaTeX Project Team. *LaTeX 2ε news 39*. June 2024. https://latex-project.org/news/latex2e-news/ltnews39.pdf