

L^AT_EX News

Issue 44, November 2026 — DRAFT version for upcoming release (L^AT_EX release 2026-11-01)

Contents

Introduction	1
News from the Tagged PDF project	1
Tags for paragraphs	1
Provide an order key mechanism for templates	2
Use the order key mechanism for theorem-like environments and proofs	2
Fix a color leakage in theorem-like environments	2
Revisiting the para-handling logic for block environments	2
Replaced legacy-code key in blockenv template	2
Make <code>\item[]</code> produce an empty item label . .	3
Use the order key mechanism for heading commands	3
Support force-unnumbered key in heading templates	3
Support placement=ragged in heading templates	3
Heading templates now support column spanning	3
Heading template instances for AMS classes . .	3
Package fncychap is now tagging compatible . .	3
<code>\newlist</code> failed for description lists	4
Storing more titles for named references	4
New or improved commands	4
Testing if a certain template is defined	4
Code improvements	4
Support for <code>\UseName</code> in template key bindings	4
Further removal of “fake math” from the kernel	4
Bug fixes	4
Avoid problems with page breaks in the middle of verbatim-like environments . . .	4
Support nested requests for class or package target dates	4
Changes to packages in the amsmath category	4
Drop special handling for Lua _T _E _X in <code>\genfrac</code>	4
Documentation changes	4
Tagging L ^A T _E X News	4
US Spelling	5

Introduction

to write

News from the Tagged PDF project

Point to talks about accessibility given at Calgary

Tags for paragraphs

Most of the time a *paragraph* just consists of a number of sentences without any further structure. However, it is also possible that it has a more complicated structure, e.g., it may contain a quote or a list with some text before or after, all belonging to the same paragraph. For example

```
Don Knuth sends a warm welcome from afar
and writes:
\begin{quote}
  \sffamily Happy TeX'ing in Calgary
\end{quote}
This is what we did!
```

To model this in the PDF structure, L^AT_EX surrounds the whole paragraph with a `<semantic-para>` tag and inside individual text blocks with their own tags, i.e., in the above example the text `Don ... writes:` and the text `This is ...!` each with a `<text-block>` tag and the quote with a `<quote>` tag¹, so you end up with

```
<semantic-para>
<text-block>
  Don Knuth sends a warm welcome from afar
  and writes:
</text-block>
<quote>
<text-block>
  Happy TeX'ing in Calgary
</text-block>
</quote>
<text-block>
  This is what we did!
<text-block>
<semantic-para>
```

Initially, we called these paragraph-related tags `<text-unit>` (now `<semantic-para>`) and `<text>` (now `<text-block>`) but the original names were difficult to understand, so we changed them now.

¹Note that HTML does not have the context of a semantic paragraph and would mark the text before and after the quote each with a `<P>` and thus effectively claiming that they are individual paragraphs while in reality they form together a single paragraph.

We also introduced the `<text-fragment>` tag for use inside `<text-block>`s if there is a need for a sub-structure.

Provide an order key mechanism for templates

Templates often have to typeset several items of textual data (from keys and/or from template arguments) with slight variations from instance to instance, e.g., the ordering and the separation between the items might differ or in other cases only a subset of the items are present (or supported). For example, a theorem-like environment typically wants to have a fixed title, a number, some punctuation, and possibly a note (provided through document input). However, one design might ask for “Lemma 3.2 (note)”, the next for “3.2 Lemma. (note)”, and another for “3.2 Lemma *note*.” with a different ordering and a punctuation somewhere or not.

We have now provided code to support this in a general way, e.g., templates can offer an “order” key in which the designer specifies the data items and their separation. Tagging support is automatically provided. The documentation is currently available via `texdoc latex-lab-template`. This will be moved to `ltemplate` in the kernel in due course.

Use the order key mechanism for theorem-like environments and proofs

The theorem-like templates have been rewritten to make use of the new `order` key mechanism. As a result a number of keys got replaced with new (more standardized) ones. Thus, for `title`, `number`, `note`, and `punct` there are now keys for customization with the names `title-decls`, `number-decls`, etc., as well as `title-format`, `number-format` and so on that expect an argument.

The default for the `order` key in theorem-like environments is

```
order={title,separator-1,number,
       separator-2,note,punct}
```

To put the number first or the punctuation directly after the number simply switch the names in the comma list around.

Also note that two distinct separator items are supported (both defaulting to a space), which means that one can easily alter one or both spaces individually simply by setting the keys `separator-1` and/or `separator-2` on the instance.

The default for `order` in proofs is slightly simpler because proofs aren’t numbered so that the template has no `number`, `number-decls` or `number-format` keys and consequently there can be at most a single separator:

```
order={title,separator,note,punct}
```

This also means proofs can now have a note (which is an extension to `amsthm`) which could be given as

```
\begin{proof}[note={my note}]
```

By default the setting of the `note-format` key surrounds the note with parentheses.

Fix a color leakage in theorem-like environments

If the `body-decls` key was given a color specification then that color leaked back into the caption of that theorem, because placing the caption was delayed until a paragraph got started by which time the body declarations were already in force.² This was fixed as part of the `order` key rewrite. (*tagging-project issues 1225 and 1350*)

Revisiting the para-handling logic for block environments

In $\text{\LaTeX}$, block environments like `itemize` or `quote` typically scan for a following empty line and if there is none then text following the environment is expected to belong to the current semantic paragraph and continues it, e.g., gets no paragraph indentation or anything else that signals the start of a new semantic paragraph.

In some situations or for some types of block environments (such as theorem-like environments) this behavior is not wanted. We have therefore added a new boolean template key `standalone` which if set to `true` causes the environment to surround itself with implicit `\par` commands mimicking the effect of explicit empty lines in the source.

We took the opportunity to also fix a couple of subtle problems with the mechanism so that it hopefully works better in a few unusual situations.

(*tagging-project issues 1402 and 1415 among others*)

Replaced legacy-code key in blockenv template

The `blockenv` template had a `legacy-code` key, used to support some legacy setup for the generic `list` environment and for the `verbatim` extension offered by the `ltugboat` class where you can write `\begin{verbatim}[\small]` to get a verbatim display in a smaller font size.

However, if the `verbatim` was directly preceded by text the content of the key was evaluated while still being in horizontal mode. In that case the `\small` changed not only the verbatim display but, as a side effect, also altered the `\baselineskip` of the preceding text. This has been corrected by providing two keys: `early-legacy-code` which is evaluated near the start of the template, possibly still in horizontal mode (used by `list`) and `late-legacy-code` which is evaluated after returning to vertical mode (used by `verbatim`).

(*tagging-project issue 1503*)

²Color in $\text{\TeX}$ is different to, say, font attributes: if you have already placed some text in a box it still reacts to the current color when the box is placed into the document. In contrast, font changes would have no effect whatsoever at this point.

Make `\item[]` produce an empty item label

After switching to key/value support in `\item` a side effect was that an empty optional argument was understood as an empty key/value list and not as a request for replacing the generated item label with an empty one. This has now been changed for better compatibility with existing usage. [\(tagging-project issue 1557\)](#)

Use the order key mechanism for heading commands

In L^AT_EX release 2026-06-01 we provided a reimplementation of heading commands using templates. The initial set of templates had only limited flexibility, modeling layouts closely related to those from the standard L^AT_EX classes. As a consequence, they used keys such as `number-title-sep` to define the space between heading number and title on the assumption that both are placed exactly in that order.

We have now written new extended the templates to allow for reordering of heading elements and more flexible spacing between them, but in that scenario such keys do no longer make sense and therefore have been replaced with more generic keys such as `separator-a`, `separator-b`, etc. not taking a length value but accepting general code.

If you have defined instances using the first template version then you can (for now) continue to use them simply by changing the template names in your instance declarations from `<name>` to `<name>-v1`, e.g., you have to change from

```
\DeclareInstance
  {heading}{chapter}{display} {...}
\DeclareInstance{headformat}
  {heading}{chapter}{display} {...}

\DeclareInstance{heading}
  {section}{hang} { ... }
\DeclareInstance{headformat}
  {section}{hang} { ... }
...
to
\DeclareInstance
  {heading}{chapter}{display-v1} {...}
\DeclareInstance{headformat}
  {heading}{chapter}{display-v1} {...}

\DeclareInstance{heading}
  {section}{hang-v1} { ... }
\DeclareInstance{headformat}
  {section}{hang-v1} { ... }
...
```

No other changes are necessary for now. However, going forward you should switch to the extended templates because the `-v1` templates will eventually be dropped.

How to upgrade to the new templates is explained in `texdoc latex-lab-sec-template`.

Support `force-unnumbered` key in heading templates to write [\(tagging-project issue 1473\)](#)

Support `placement=ragged` in heading templates

The `placement` key in heading templates was augmented to support the value `ragged` in addition to `page`, `top`, and `normal`. This key value works like `normal` (i.e., the heading can appear anywhere on the page) but if the heading happens to trigger a page break then the previous page will be set “raggedbottom” and not spread out to fill the page in the `book` or `report` classes. This is useful on pages that do not have a lot of stretchability in the first place and thus the space between paragraphs is used for stretching which often looks ugly.

It can either be specified on the instance level by setting the `placement` key to `ragged` or for individual headings in the document by setting it in the optional argument to the heading. [\(github issue 1844\)](#)

Heading templates now support column spanning

In twocolumn documents top-level headings usually start on a new page and span both columns. In L^AT_EX 2_ε this was hardwired and headings such as `\chapter` showed this behavior. In the new template-based implementation this is now more flexible and one sets this independently from forcing a new page by setting the key `column-spanning` to `true` (default for `\chapter`) or `false` (default for other headings). Thus

```
\section[column-spanning=true]{title}
```

generates a one-off `\section` heading that spans both columns. Due to the way the L^AT_EX output routine is currently implemented that automatically also starts a new page, i.e., one can’t generate spanning headings in the middle of a page (that can only be done with the `multicol` package at the moment). That restriction may be lifted when the output routine is redefined.

[\(tagging-project issue 1436\)](#)

Heading template instances for AMS classes

to write, including `\AddPunct`

[\(tagging-project issue 1477\)](#)

Package `fncychap` is now tagging compatible

The `fncychap` packages defines half a dozen chapter layouts that can be selected through options to the package. The layouts can then be further customized through a series of commands defined by the package. Due to its age and the fact that it overwrote various internals, the resulting chapter headings were not usable when producing a tagged and accessible document.

We have now provided a simple template with appropriate tagging support (but without any customization

keys) that makes use of the code in the package and this way supports all the heading designs offered by the package as well as the customization possibilities provided by the package. With these fairly minor adjustments the package is now compatible with tagging and can be used when making accessible documents. Because of the customization possibilities that allow for arbitrary rewrites we have to make the tagging support fairly general to cover such variants. There is a downside to this: in several designs we end up with a number of empty `heading-fragment` structures. While this is allowed and doesn't do any harm (other than resulting in some unnecessary processing by the PDF reader) it is not as elegant as a solution with standard templates would be. *(tagging-project issue 570)*

`\newlist` failed for description lists

The emulation of `enumitem`'s `\newlist` declaration failed when the source list was `description`. This has now been corrected. *(tagging-project issue 1518)*

Storing more titles for named references

When using the `nameref` package it is possible to reference the names of theorems (as given by the optional argument) and to reference the optional argument of items of description lists. `nameref` made this possible by patching the internal commands of theorems and lists. These patches no longer worked with the new theorem and list code when `\DocumentMetadata` is used. This has now been repaired: Theorems and items with optional arguments will update `\@currentlabelname` and so allow `\label` to store this value for later reference. Items will also store the optional argument in `\@currentlabel` and so `\ref` will reference this value:

```
\begin{enumerate}
\item First item
\item[1b] \label{item}
\end{enumerate}
\ref{item} % gives now 1b
```

(tagging-project issue 1538)

New or improved commands

Testing if a certain template is defined

To better support the declaration of template instances we now provide the conditional `\IfTemplateExistsTF` (and its T and F variants) so that it is possible to take different actions depending on whether or not a certain template is declared.

Code improvements

Support for `\UseName` in template key bindings

We have extended the logic in the key-binding argument for `\DeclareTemplateCode`, such that you can use the keyword `name` here (in addition to `global`) without any

performance impact when using templates. This allows automatic variable creation with “awkward” characters, most notably – and numerals. Of course, such variables then also need to be referred to via `\UseName` or some equivalent method in the code, so this is only useful if there is a pressing need for such special names.

Further removal of “fake math” from the kernel

In the previous release, we removed most use of “fake math” (math mode not denoting a formula but used for positioning or manipulating text material) from the kernel [1]. The use of `\underline` in text mode was missed in that update: this has now been addressed.

Bug fixes

Avoid problems with page breaks in the middle of verbatim-like environments

The fix we announced in L^AT_EX News 41 [1, p.116] was incomplete: we missed out resetting the `\catcode` of comma. This was now corrected. *(github issue 2056)*

Support nested requests for class or package target dates

If one loads a package with `\usepackage{<package>}[=<date>]` then this means the package version current at `<date>` should be loaded. For document classes `\documentclass` offers the same interface. Inside a package or a class a developer can produce conditional code that depends on the requested target date (if any) by using the command `\IfTargetDateBefore`, see [2] for an article on this functionality.

Unfortunately, using the the optional argument on one package had effects on the behavior of `\IfTargetDateBefore` in all following code, i.e., it was not confined to that package code for which this request was made. This has now been corrected.

(github issue 828)

Changes to packages in the `amsmath` category

Drop special handling for Lua_TE_X in `\genfrac`

The `\genfrac` command relies on engine functionality to place delimiters around fractions in traditional T_EX engines but emulated this functionality in X_YT_EX and Lua_TE_X. For Lua_TE_X this emulation was broken and is not needed and interferes with tagging, therefore it has been removed. *(github issue 432)*

Documentation changes

Tagging L^AT_EX News

We have adjusted the sources for *L^AT_EX News* (including this news!) to enabling tagging (PDF/UA-2) automatically. This applies to the combined newsletters and each individual file. This is part of our effort to make all L^AT_EX documentation accessible over time.

US Spelling

We have (finally) decided that standardizing the spelling in the L^AT_EX sources to use US-English is the realistic position.³ This means that the documentation better aligns with the code. Note that idiom may still be more varied: this is harder to adjust than spelling.

References

- [1] L^AT_EX Project Team. *L^AT_EX 2_ε news 1–44*. November, 2026. <https://latex-project.org/news/latex2e-news/ltnews.pdf>
- [2] Frank Mittelbach *A rollback concept for packages and classes*. *TUGboat* 39:2, 2018. <https://latex-project.org/publications/2018-FMi-TUB-tb122mitt-version-rollback.pdf>

³None of the native English speakers on the team are from the US!