

L^AT_EX News

Issue 35, June 2022 — DRAFT version for upcoming release

Contents

Introduction	1
Document metadata interface	1
The latex-lab bundle	2
New or improved commands	2
A keyval approach to option handling	2
Floating point and integer calculations	2
CamelCase commands for changing arguments	
to csnames	2
???	3
Code improvements	3
A small update to <code>\obeylines</code> and	
<code>\obeyspaces</code>	3
<code>ltxdoc</code> gets a <code>nocfg</code> option	3
<code>doc</code> upgraded to version 3	3
<code>doc</code> can now show dates in change log	3
Lua _T _E X callback improvements	3
Class <code>proc</code> supports <code>twoside</code>	3
Croatian character support	3
Cleanup of the Unicode declaration interface .	4
???	4
Bug fixes	4
Using <code>\DeclareUnicodeCharacter</code> with C1	
control points	4
Fix <code>\ShowCommand</code> when used with <code>ltxcmd</code> . . .	4
Changes to packages in the amsmath category	4
<code>amsopn</code> Do not reset <code>\operator@font</code>	4
???	4
Changes to packages in the graphics category	4
Color in formulas	4
???	4
Changes to packages in the tools category	4
<code>multicol</code> : Fix <code>\newcolumn</code>	4
<code>bm</code> : Fix for amsmath operators	5

Introduction

Document metadata interface

Until recently there was no dedicated location to declare settings that affect a document as a whole. Settings

had to be placed somewhere in the preamble or as class options or sometimes even as package options. For some such settings this may cause issues, e.g., setting the PDF version is only possible as long as the PDF output file has not yet been opened which can be caused by loading one or the other package.

For the “L^AT_EX Tagged PDF project” [1] further metadata about the whole document (and its processing) need to be specified and again this data should be all placed in a single well-defined place.

For this reason we introduce the new command `\DocumentMetadata` to unify all such settings in one place. The command expects a key/value list that describes all document metadata for the current document. It is only allowed to be used at the very beginning of the document, i.e., the declaration has to be placed *before* `\documentclass` and will issue an error if found later.

At this point in time we only provide the bare command in the format, the actual processing of the key/value is defined externally and the necessary code will be loaded if the command is used. This scheme is chosen for two reasons: by adding the command in the kernel it is available to everybody without the need to load a special package using `\RequirePackage`. The actual processing, though, is external so that we can easily extend the code (e.g., offering additional keys or changing the internal processing) while the above mentioned project is progressing. Both together allows users to immediately benefit from intermediate results produced as part of the project, as well as offering the L^AT_EX Project Team the flexibility to enable such intermediate results (for test purposes or even production use) in-between and independently of regular L^AT_EX releases. Over time, tested and approved functionality can then seamlessly move into the kernel at a later stage without any alterations to documents already using it. At the same time, not using the new consolidated interface means that existing documents are in no way affected by the work that is carried out and is in a wider alpha or beta test phase.

Documentation about the new command and already existing keys are in `l3meta.pdf` and `documentmetadata-support.pdf` and also in the documentation of the `pdfmanagement-testphase` package.

The latex-lab bundle

write about it

New or improved commands

A keyval approach to option handling

The classical L^AT_EX 2_ε method for handling options, using `\ProcessOptions`, treats each entry in the list as a string. Many package authors have sought to extend this handling by treating each entry as a key–value pair (keyval) instead. To-date, this has required the use of additional packages, for example `kvoptions`.

The L^AT_EX team have for some time offered the package `l3keys2e` to allow keyvals defined using the L3 programming layer module `l3keys` to act as package options. This ability has now been integrated directly into the kernel. As part of this integration, the syntax for processing keyval options has been refined, such that

```
\ProcessKeyOptions
```

will now automatically pick up the package name as the key *family*, unless explicitly given as an optional argument.

```
\ProcessKeyOptions[family]
```

A version which does not consider global options, `\ProcessKeyPackageOptions`, is also available.

To support creating key options in for this mechanism, the new command `\DeclareKeys` has been added. This works using the same general approach as `l3keys` or `pgfkeys`: each key has one or more *properties* which define its behavior.

Options for packages which use this new approach will not be checked for clashes by the kernel. Instead, each time a `\usepackage` or `\RequirePackage` line is encountered, the list of options given will be passed to `\ProcessKeyPackageOptions`. Options which can only be given the first time a package is loaded can be marked using the property `.usage = load`, and will result in a warning if used in a subsequent package loading line.

Package options defined in this way can also be set within a package using the new command `\SetKeys`, which again takes an optional argument to specify the *family*, plus a mandatory one for the options themselves.

Floating point and integer calculations

The L3 programming layer offers expandable commands for calculating floating point and integer values, but so far these functions have only been available to programmers, because they require `\ExplSyntaxOn` to be in force. To make them easily available at the document-level, the small package `xfp` defined `\fpeval` and `\interval`.

An example of use could be the following:

```
\LaTeX{} can now compute:
```

```
\[ \frac{\sin (3.5)}{2} + 2\cdot 10^{-3}
```

```
= \fpeval{\sin(3.5)/2 + 2e-3} \]
```

which produces the following output:

L^AT_EX can now compute:

$$\frac{\sin(3.5)}{2} + 2 \cdot 10^{-3} = -0.1733916138448099$$

These two commands have now been moved into the kernel and in addition we also provide `\dimeval` and `\skipeval`. The details of their syntax are described in `usrguide3.pdf`. The command `\fpeval` offers a rich syntax allows for extensive calculations whereas the other three commands are essentially thin wrappers for `\numexpr`, `\dimexpr`, and `\glueexpr`—therefore inheriting some syntax peculiarities and limitations in expressiveness.

```
\newcommand\calculateheight[1]{%
  \setlength\textheight{\dimeval{\topskip
    + \baselineskip * \interval{#1-1}}}
```

The above, for example, calculates the appropriate `\textheight` for a given number of text lines.

(github issue 711)

CamelCase commands for changing arguments to csnames

It is sometimes helpful to “construct” a command name on the fly rather than providing it as a single `\dots` token. For these kind of tasks the L^AT_EX 3 programming layer offers a general mechanism (in form of `\exp_args:N...` and `\cs_generate_variant:Nn`). However, when declaring new document-level commands with `\NewDocumentCommand` or `\NewCommandCopy`, etc. the L3 programming layer may not be active, and even if it is, mixing CamelCase syntax with L3 programming syntax is not really a good approach. We have therefore added the commands `\UseName` and `\ExpandArgs` to assist in such situations, e.g.,

```
\NewDocumentCommand\newcopyedit{m0{red}}
  {\newcounter{todo#1}%
   \ExpandArgs{c}\NewDocumentCommand{#1}{s m}%
   {\stepcounter{todo#1}%
    \IfBooleanTF {##1}%
    {\todo[color=#2!10}%
     {\UseName{thetodo#1}: ##2}}}%
   {\todo[inline,color=#2!10}%
    {\UseName{thetodo#1}: ##2}}}%
  }%
```

which provides a declaration mechanism for copyedit commands, so that `\newcopyedit{FMi}[blue]` then defines `\FMi` (and the necessary counter).

The command `\ExpandArgs` can be useful with the argument `cc` or `Nc` in combination with `\NewCommandCopy`

if the old or new command name or both need constructing. Finally, there is `\UseName` which takes its argument and turns it into a command (i.e., a CamelCase version of `\@nameuse` (L^AT_EX 2_ε) or `\use:c` (L3 programming layer)) which was also used in the example above.

(github issue 735)

???

(github issue ???)

Code improvements

A small update to `\obeylines` and `\obeyspaces`

The plain T_EX versions of `\obeylines` and `\obeyspaces` make `^^M` and `␣` active and force them to execute `\par` and `\space`, respectively. Don Knuth makes a remark in the T_EXbook that one can then use a trick such as

```
\let\par=\cr \obeylines \halign{...
```

However, redefining `\par` like this is not really a great idea in L^AT_EX, because it may lead to all kind of problems. We have therefore changed the commands to use an indirection: the active characters now execute `\obeyedline` and `\obeyedspace`, which in turn do what the hardwired solution did before.

```
But • this • means • that • it • is • now • possible
    • to • achieve • special • effects • in • a • safe •
way. • This • paragraph, • for • example, • was •
produced • by • making • \obeyedspace • generate
    • {\␣\textbullet␣} • and • enabling •
\obeyspaces • within • a • quote • environment.
```

Thus, if you are keen to use the plain T_EX trick, you need to say `\let\obeyedlines=\cr` now. (github issue 367)

`ltxdoc` gets a `nocfg` option

The L^AT_EX sources are formatted with the `ltxdoc` class, which supports loading a local config file `ltxdoc.cfg`. In the past the L^AT_EX sources used such a file but it was not distributed. As a result reprocessing the L^AT_EX sources elsewhere showed formatting changes. We now distribute this file which means that it is loaded by default. With the option `nocfg` this can be prevented.

`doc` upgraded to version 3

After roughly three decades the `doc` package gets a cautious uplift, as already announced at the TUG conference 2019—changes to `doc` are obviously always done in a leisurely manner.

Given that most documentation is nowadays viewed on screen, `hyperref` support is added and by default enabled (suppress it with option `nohyperref` or alternatively with `hyperref=false`) so the internal cross-reference are properly resolved including those from the index back into the document.

Furthermore, `doc` has now a general mechanism to define additional “doc” elements besides the two `Macro`

and `Env` it did know in the past. This enables better documentation because you can now clearly mark different types of objects instead of simply calling them all “macros”. If desired, they can be collected together under a heading in the index so that you have a section just with your document interface commands, or with all parameters, or ...

The code borrows ideas from Didier Verna’s `dox` package (although the document level interface is different) and it makes use of Heiko Oberdiek’s `hypdoc` package, which at some point in the future will be completely integrated, given that its whole purpose is to patch `doc`’s internal commands to make them `hyperref`-aware.

All changes are expected to be upward compatible, but if you run into issues with older documentation using `doc` a simple and quick solution is to load the package as follows: `\usepackage{doc}[=v2]`

`doc` can now show dates in change log

Up to now the change log was always sorted by version numbers (ignoring the date that was given in the `\changes` command). It can now be sorted by both version and date if you specify the option `reportchangedates` on package level and in that case the changes are displayed with

`⟨version⟩ – ⟨date⟩`

as the heading (instead of just `⟨version⟩`), when using `\PrintChanges`. (github issue gh/531)

LuaT_EX callback improvements

The LuaT_EX callbacks `hpack_quality` and `vpack_quality` are now `exclusive` and therefore only allow a single handler. The previous type `list` resulted in incorrect parameters when multiple handlers were set, therefore this only makes an existing restriction more explicit.

Additionally the return value `true` for `list` callbacks is now handled internally and no longer passed on to the engine. This simplifies the handling of these callbacks and makes it easier to provide consistent interfaces for user defined `list` callbacks.

Class `proc` supports `twoside`

The document class `proc`, which is a small variation on the `article` class, now supports the `twoside` option displaying different data in the footer line on recto and verso pages. (github issue gh/704)

Croatian character support

The default `inputenc` support has been extended to support the 9 characters DŽ, Dž, dž, LJ, Lj, lj, NJ, Nj, nj, input as single UTF-8 codepoints in the range U+01C4 to U+01CC. (github issue gh/723)

Cleanup of the Unicode declaration interface

When declaring encoding specific commands for the Unicode (TU) encoding some declarations (e.g., `\DeclareUnicodeComposite`) do not have an explicit argument for the the encoding name, but instead use the command `\UnicodeEncodingName` internally. There was one exception though: `\DeclareUnicodeAccent` required an explicit encoding argument. This inconsistency has now been removed and the encoding name is always implicit. To avoid a breaking change for a few packages on CTAN `\DeclareUnicodeAccent` still accepts three arguments if the second argument is TU or `\UnicodeEncodingName`. Once all packages have been updated this code branch will get removed.

At the same time we added `\DeclareUnicodeCommand` and `\DeclareUnicodeSymbol` for consistency. They also use `\UnicodeEncodingName` internally, instead of requiring an encoding argument as their general purpose counterparts do.

(github issue 253)

???

(github issue ???)

Bug fixes

Using `\DeclareUnicodeCharacter` with C1 control points

An error in the UTF-8 handling for non-Unicode T_EX, has prevented `\DeclareUnicodeCharacter` being used with characters in the range hex 80 to 9F, this has been corrected in this release.

(github issue 730)

Fix `\ShowCommand` when used with `ltxcmd`

When `\ShowCommand` support was added for `ltxcmd` in the previous release [6], a blunder in the code made it so that when `\ShowCommand` was used on a command defined with `ltxcmd`, it only printed the meaning of the command in the terminal, but didn't stop for interaction as it does elsewhere (mimicking `\show`). The issue is now fixed.

(github issue 739)

Changes to packages in the *amsmath* category

amsopn Do not reset `\operator@font`

The package `amsopn` used to define `\operator@font` but this command is already provided by the L^AT_EX format (for at least 14 years). As a result the definition in `amsopn` is equivalent to a reset to the kernel definition, which is unnecessary and suprising if you alter the math setup (e.g., by loading a package) and at a later stage add `amsmath`, which then undoes part of your setup. For this reason the definition was taken out and `amsmath/amsopn` now relies on the format definition.

In the unlikely event that you want the resetting to happen, use

```
\makeatletter
\def\operator@font{\mathgroup\symoperators}
```

```
\makeatother
```

after loading the package.

(github issue 734)

???

(github issue ???)

Changes to packages in the *graphics* category

Color in formulas

While it is possible to color parts of a formula using `\color` commands the approach is fairly cumbersome. For example, to color an summation sign, but not its limits, you need four `\color` commands and some seemingly unnecessary set of braces to get coloring and spacing right:

```
\[ X = \color{red} \sum
% without {{ the superscript below is misplaced
% without {{ the \sum is black
\color{black} % without it the x_i is red
x_i \]
```

Leave out any of the `\color` commands or any of the `{{...}}` will give you a wrong result instead of the desired

$$X = \sum_{i=1}^n x_i$$

So even if this is possible, it is not a very practical solution and furthermore there are a number of cases where it is impossible to color a certain part of a formula, for example, an opening symbol such as `\left(` but not the corresponding `\right)`.

We have therefore added the command `\mathcolor` to the `color` and `xcolor` package, which has the same syntax as `\textcolor`, but is specially designed for use in math and handles sub and superscripts and other aspects correctly and preserves correct spacing. Thus, the above example can now be written as

```
\[ X = \mathcolor{red}{\sum}_{i=1}^n x_i \]
```

This command is *only* allowed in formulas. For details and further examples, see `mathcolor.pdf`.

???

(github issue ???)

Changes to packages in the *tools* category

multicol: Fix `\newcolumn`

The recently added `\newcolumn` didn't work properly if used in vertical mode, where it behaved like `\columnbreak`, i.e., spreading the column material out instead running the column short.

(<https://tex.stackexchange.com/q/624940>)

bm: Fix for amsmath operators

An internal command used in the definition of operator commands such as `\sin` in `amsmath` has been guarded in `\bm` to prevent internal syntax errors due to premature expansion. *(github issue 744)*

References

- [1] Frank Mittelbach and Chris Rowley: *L^AT_EX Tagged PDF—A blueprint for a large project*.
<https://latex-project.org/publications/indexbyyear/2020/>
- [2] *L^AT_EX documentation on the L^AT_EX Project Website*.
<https://latex-project.org/help/documentation/>
- [3] L^AT_EX Project Team: *L^AT_EX 2_ε news 31*.
<https://latex-project.org/news/latex2e-news/ltnews31.pdf>
- [4] L^AT_EX Project Team: *L^AT_EX 2_ε news 32*.
<https://latex-project.org/news/latex2e-news/ltnews32.pdf>
- [5] L^AT_EX Project Team: *L^AT_EX 2_ε news 33*.
<https://latex-project.org/news/latex2e-news/ltnews33.pdf>
- [6] L^AT_EX Project Team: *L^AT_EX 2_ε news 34*.
<https://latex-project.org/news/latex2e-news/ltnews34.pdf>