

# memoize-ext: Extensions for memoize

Clea F. Rees\*

11782 2026-01-19

## Abstract

memoize-ext provides extensions for Živanović’s package memoize (2024). In particular, it supports the memoization of content in tagged PDFs and presentations produced with Wright’s ltx-talk (2026).

## Contents

<b>I Usage</b>	<b>3</b>
1 Basics	3
2 expl3	4
3 l3draw	4
4 ltx-talk	4
5 Tagged PDF	5
5.1 TikZ pictures	5
5.2 Other content	6
<b>II Implementation</b>	<b>7</b>
memoize-ext	7
memoize-ext-expl3	12
memoize-ext-l3draw	19
memoize-ext-sockets	21

---

\*Bug tracker: [codeberg.org/cfr/memoize-ext/issues](https://codeberg.org/cfr/memoize-ext/issues) | Code: [codeberg.org/cfr/memoize-ext](https://codeberg.org/cfr/memoize-ext) | Mirror: [github.com/cfr42/memoize-ext](https://github.com/cfr42/memoize-ext)

---

memoize-ext-tag . . . . .	22
memoize-ext-talk . . . . .	32

# Part I

## Usage

### 1 Basics

Usage is simple.

```
\documentclass{<class>}
\usepackage{memoize-ext}
```

The package will automatically load `memoize` and pass any unrecognised options onto that package.

Note that to create a tagged presentation with `ltx-talk`, the package should be loaded *after* the class<sup>1</sup>.

```
\DocumentMetadata{tagging=on,lang=en-GB,pdfversion=2.0,pdfstandard=UA-2,}
\documentclass{ltx-talk}
\usepackage{memoize-ext}
```

If for any reason it is necessary to load the package *prior* to the class in a tagged PDF, then tagging must be explicitly requested. For example,

```
\DocumentMetadata{tagging=on,lang=cy,pdfversion=2.0,pdfstandard=ua-2,}
\RequirePackage[tag]{memoize-ext}
\documentclass{book}
```

If necessary, a small number of package options are available to customise which code is loaded.

`expl3 (opt.) = true|false`

Loads code supporting `expl3` syntax.

Default is `true`. Initially `false`.

`l3draw (opt.) = true|false`

Loads code supporting `l3draw`, if the package is loaded.

Default is `true`. Initially `true`.

`tag (opt.) = true|false`

Loads code supporting tagged PDF, if L<sup>A</sup>T<sub>E</sub>X's tagging code is activated when the package is loaded. Note that this is *not* true prior to `\documentclass`.

Default is `true`. Initially `true` if tagging is activated; `false` otherwise.

`talk (opt.) = true|false`

Loads code supporting `ltx-talk`, if the class is loaded.

<sup>1</sup>Note that `ltx-talk` diverges from `beamer` here, a point to which I was oblivious when I wrote the initial version of this documentation.

Default is `true`. Initially `true`.

Note that the additional code is not loaded if a different class is used, regardless of this setting. The option is provided in case it is necessary to disable support for the class, without disabling other parts of `memoize-ext`.

## 2 expl3

`replicate expl fn` (*pgfkey*) Sets up advice to ‘replicate’ an `expl3` function.

This works similarly to the builtin support for commands created with `\NewDocumentCommand` etc. This means that it is not necessary to specify `args`.

**Functions with w-type arguments are NOT supported.** Attempting to use this key with such a function will result in an error. Such cases require custom handling and can be configured using the standard `memoize` keys.

`memoize-ext-l3draw.sty` demonstrates use of `replicate expl fn`:

```
\mmzset{%
  auto~csmname={draw_begin:}{memoize,collector=\AdviceCollectDrawArguments,},
  auto~csmname={_draw_record_origin:}{run~if~memoizing,replicate~expl~fn,},
}
```

This code sets up a custom ‘collector’ and installs ‘advice’ which memoizes `\draw_begin:`. It further advises `\_draw_record_origin:` so that if this function is found during memoization, it will be replicated in the `ccmemo`. This ensures that the origin is recorded correctly when the memoized picture is utilised, since we do not want to record its position when memoized.

The net result of this is that auto-memoization of `l3draw` pictures should (hopefully) ‘just work’.

## 3 l3draw

`sec:draw` «< `l3draw` pictures are auto-memoized by default. `memoize-ext-l3draw.sty` mostly exists to demonstrate use of `replicate expl fn`. »> `sec:draw`

## 4 ltx-talk

The code is based on that provided by `memoize` for `beamer` and supports the same options, except that ‘`talk`’ is substituted for ‘`bemaer`’.

`per overlay` (*pgfkey*) Equivalent to the `beamer` option of the same name.

`talk mode to prefix` Equivalent to `beamer mode to prefix`.

(*pgfkey*) The code uses and/or changes internal code from both `ltx-talk` and `memoize`. While the public interface for `memoize` is fairly stable, the internals may not be, and `ltx-talk` is highly experimental. The latter also uses a large number of experimental packages and makes extensive use of experimental `LATEX` features.

The justification for publishing this part of `memoize-ext` is essentially that anybody using `ltx-talk` and `memoize` is already playing with fire, so it is better to have an unreliable extinguisher to hand than none at all.

A few things you should know, even if you do not want to:

- the code uses an internal `ltx-talk` boolean to drive extern creation and utilisation;
- `talk mode to prefix` relies on an internal `ltx-talk` string;
- to workaround incompatibilities between `memoize` and `pdfmanagement`, the code redefines an internal `memoize` macro<sup>2</sup>.

## 5 Tagged pdf

If using the package to produce tagged PDF, note that the tagging support

- redefines the internal macro `\mmzIncludeExtern` during utilisation;
- relies on an internal string variable in `lsockets`.

**Correct tagging *requires* that unmemoizable code be marked as such, either manually or automatically.** The package does this automatically for `tikzpicture`s which use an unsupported tagging plug, but does nothing in any other case. So if your picture uses `remember picture`, for example, you *must* mark the code as unmemoizable or disable tagging for the affected code. The package will warn you about this, but that is all it does.

### 5.1 TikZ pictures

If the content you wish to memoize is a `TikZ` picture, you probably do not need to do anything special, but note that the default `latex-lab` plug is *not* supported. You must use one of `alt`, `actualtext` or `artifact`.

If you use `alt` or `actualtext` in the optional argument to `tikzpicture`, the value will be recorded in the `ccmemo` for use during utilisation. If you set the value outside the `tikzpicture`, this is not necessary. In the latter case, the *extern* will not depend on the value given (unless you request that specifically).

Note that if you change the selected plug *and* you set this *outside* the picture, you must manually tell `memoize` it should recompile the picture, since the plug is recorded in the `ccmemo`, but the hash will not have changed.

Note that tagging is disabled during memoization and additionally *disabled for content which has just been memoized*. So when a run produces an extern, the memoized code will not be tagged at all.

---

<sup>2</sup>The redefinition injects code into the box `memoize` ships out which resets opacity before and after the memoized code is executed. This is required because `memoize` relies on primitive `shipout`, whereas the implementation of opacity in `pdfmanagement` relies on  $\LaTeX$ 's `shipout` routine.

**Note that this package does *not* support forest.** If your document uses `forest` (Živanović 2017), you should either disable memoization for these pictures or load `forest-ext`<sup>3</sup> (Rees 2026).

The `TikZ` support is implemented by replacing plugs provided by `latex-lab` with versions designed for memoized content (L<sup>A</sup>T<sub>E</sub>X Project 2025b). Code is also installed into the same hooks `latex-lab` uses with rules to ensure this package's has priority.

`mmzx (plug)` Plug for `tagsupport/tikz/picture/init`

If memoization is not active, the plug executes the `latex-lab default` plug.

If some option for this package is specifically configured, it is used. Otherwise, the code initialisation code at the start of the picture attempts to find a match for any configured `latex-lab` plug. In effect, this means that you should not need to change anything in your document if you use one of the three supported plugs.

If memoization is enabled but no suitable plug is found, a warning is issued and memoization aborted. Otherwise, code is inserted into the `ccmemo` to emulate the appropriate `latex-lab` plug. In most cases, this code simply calls the relevant `latex-lab` plugs.

Plugs for `tagsupport/tikz/picture/begin` and `tagsupport/tikz/picture/end`:

`mmzx/actualtext (plug)` Sets up the `ccmemo` to use the `latex-lab actualtext` plugs.

`mmzx/artifact (plug)` Sets up the `ccmemo` to use the `latex-lab artifact` plugs.

`mmzx/alt (plug)` This pair of plugs is the exception. Rather than writing a `ccmemo` which will invoke the `latex-lab alt` plugs, these plugs write a `ccmemo` which uses an alternative implementation of those plugs. The reimplementation uses *properties* (provided by the L<sup>A</sup>T<sub>E</sub>X format) rather than *rememberpicture* (provided by PGF/TikZ)<sup>4</sup>.

*If everything looks OK, tagging is disabled for the current picture.* This is efficient if memoization is successful, but may be problematic if memoization is aborted or fails. In this case, it may be necessary to mark the content as unmemoizable or to disable memoization for particular pictures, in order to ensure content is tagged correctly<sup>5</sup>.

## 5.2 Other content

If the content you wish to memoize is *not* a `TikZ` picture, you may need to read the remainder of this section.

Generic support is provided in the form of two sockets which are used directly before and directly after an extern is included during utilisation. By default, the sockets do nothing, but they may be used to inject code which wraps the included extern in a suitable tagging structure.

Plugs may be assigned to the sockets either by writing suitable code to the `ccmemo` or in the document itself. The `TikZ` support, for example, writes commands to the `ccmemo` which assign plugs analogous to the `latex-lab` plugs available for non-memoized pictures.

<sup>3</sup>This is not necessary if you use `prooftrees`, which will load the package automatically if required.

<sup>4</sup>I considered using the support provided for `\includegraphic`, but this would require more intrusive changes to the internals of `memoize` and would essentially duplicate bounding box calculations already completed during memoization.

<sup>5</sup>It would be possible to disable tagging only if memoization succeeds, but I am not sure whether the structure will be right in this case?

tagsupport/memoize/include/extern/before

(*socket*) This socket receives three arguments during extern utilisation: the width, height and depth of the memoized content. The `alt` plug for `TikZ`, for example, uses these values to calculate the bounding box required to create a `Figure` structure with `alt` text.

This socket is used just before the extern is included in the document.

tagsupport/memoize/include/extern/after

(*socket*) This socket absorbs no arguments. During extern utilisation, it is used immediately after inclusion of an extern.

## Part II

# Implementation

A double underscore (`__`) or an ‘at’ (`@`) indicates an internal macro or key. These are liable to change without notice and should not be used elsewhere. Some additional macros are categorised in the same way, but are named differently to simplify use in memos<sup>6</sup>.

## memoize-ext

```
<*sty> <@@=mmzx>
```

```
1 \NeedsTeXFormat{LaTeX2e}[2021-11-15]%
```

copied verbatim, excepting format from Joseph Wright’s `siunitx.sty` under LPPL

```
2 \@ifundefined{ExplLoaderFileDate}{%
3 \RequirePackage{expl3}%
4 }{}
```

almost verbatim from `siunitx.sty`

should check date requirement (copied from `chronos`)

```
5 \@ifl@t@r\ExplLoaderFileDate{2022-02-24}{%
6 }{%
7 \PackageError{memoize-ext}{Support package expl3 too old}
8 {%
9 You need to update your installation of the bundles 'l3kernel' and
10 'l3packages'.\MessageBreak
11 Loading memoize-ext will abort!%
12 }%
13 \endinput
14 }%
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16 \GetIdInfo $Id: memoize-ext.dtx 11782 2026-03-17 14:49:00Z cfrees $ {Extensions for
17 <|debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
18 <|debug> {v0.3.4 \ExplFileVersion}{\ExplFileDescription}
```

<sup>6</sup>This follows `memoize`’s own practice.

```

19 <debug>    \ProvidesExplPackage{\ExplFileName-debug}
20 <debug>    {\ExplFileDate}{v0.3.4 \ExplFileVersion}{\ExplFileDescription}
21 %
22 \str_new:N \g__mmzx_name_str
23 \str_gset:NV \g__mmzx_name_str \ExplFileName
24 %
25 <!debug>   \disable@package@load {memoize-ext-debug}
26 <debug>    \disable@package@load {memoize-ext}
27 {
28   Only one of memoize-ext and memoize-ext-debug should be loaded.
29   Since
30 <!debug>   memoize-ext
31 <debug>    memoize-ext-debug
32   had been loaded,I will ignore your request for
33 <debug>    memoize-ext
34 <!debug>   memoize-ext-debug
35 .}
36 \SetDefaultHookLabel{memoize-ext}

```

`\l__mmzx_opt_tag_bool` (*var.*) Set according to activation status by default.

```

37 \bool_new:N \l__mmzx_opt_tag_bool
38 \tag_if_active:TF
39 { \bool_set_true:N \l__mmzx_opt_tag_bool }
40 { \bool_set_false:N \l__mmzx_opt_tag_bool }

```

`\l__mmzx_opt_draw_bool` (*var.*) Other bools.

`\l__mmzx_opt_expl_bool` (*var.*)

```

41 \keys_define:nn {memoize-ext}
42 {
43 <!*debug>
44   debug      .code:n      = {
45     \PackageWarning{memoize-ext}{
46       To load the debugging code,use memoize-ext-debug instead of this package.
47     }
48   },
49 </!debug>
50   expl3      .bool_set:N = \l__mmzx_opt_expl_bool,
51   expl3      .default:n  = true,
52   expl3      .initial:n  = false,
53   l3draw     .bool_set:N = \l__mmzx_opt_draw_bool,
54   l3draw     .default:n  = true,
55   l3draw     .initial:n  = true,
56   tag        .bool_set:N = \l__mmzx_opt_tag_bool,
57   tag        .default:n  = true,
58   talk       .bool_set:N = \l__mmzx_opt_talk_bool,
59   talk       .default:n  = true,
60   talk       .initial:n  = true,
61 }
62 \DeclareUnknownKeyHandler{
63   \PassOptionsToPackage{\CurrentOption}{memoize}
64 }

```

`\IfFormatAtLeastTF` Joseph Wright: from siunitx.sty ; <https://chat.stackexchange.com/transcript/message/>

64327823#64327823

```

65 \providecommand \IfFormatAtLeastTF { \@ifl@t@r \fmtversion }

66 \IfFormatAtLeastTF { 2022-06-01 }
67 {
68   \ProcessKeyOptions [ memoize-ext ]
69 }{
70   \RequirePackage { l3keys2e }
71   \ProcessKeysOptions { memoize-ext }
72 }

73 \IfFormatAtLeastTF { 2020-10-01 }{
74 }{
75   \RequirePackage { xparse }
76   \providecommand \ExpandArgs [1]
77   { \cs_if_exist_use:c { exp_args:N #1 } }
78 }

```

Should specify next version here, most probably. Or conditionalise input switch for ccmemos?

```

79 \RequirePackage{memoize}
80 <debug>   \mmzset{
81 <debug>     trace,
82 <debug>     include context in ccmemo,
83 <debug>   }

```

Fix for \toksapp and friends courtesy of Max Chernoff <https://github.com/sasozivanovic/memoize/pull/57/commits>.

```

84 \sys_if_engine luatex:F
85 {
86   \clist_map_inline:nn {\toksapp,\etoksapp,\gtoksapp,\xtoksapp}
87   {
88     \tl_if_head_eq_meaning:VNF #1 \protected
89     {
90       \expandafter\protected\expandafter\def\expandafter#1\expandafter{#1}
91     }
92   }
93 }

```

temporary variables, quarks

```

94 \bool_new:N \l__mmzx_tmpa_bool
95 \fp_new:N \l__mmzx_tmpa_fp
96 \int_new:N \l__mmzx_tmpa_int
97 \quark_new:N \q__mmzx_stop
98 \tl_new:N \l__mmzx_tmpa_tl
99 \tl_new:N \l__mmzx_tmpb_tl
100 \tl_new:N \l__mmzx_tmpc_tl
101 \seq_new:N \l__mmzx_tmpa_seq
102 \str_new:N \l__mmzx_tmpa_str
103 \str_new:N \l__mmzx_tmpb_str
104 <*debug>
105 \cs_new_protected:Npn \__mmzx_debug:n #1
106 {

```

```

107 \iow_log:n {[mmzx debug]:: #1}
108 }
109 \cs_generate_variant:Nn \__mmzx_debug:n {e}
110 \cs_new_protected:Npn \__mmzx_debug:N #1
111 {
112   \__mmzx_debug:e {\cs_to_str:N #1: \exp_args:NV \exp_not:n #1}
113 }
114 </debug>

```

\\_\_mmzx\_noop: Do nothing successfully.

```

\__mmzx_noop:n
115 \cs_new:Npn \__mmzx_noop: {}
116 \cs_new:Npn \__mmzx_noop:n {}

```

tag, expl, l3draw, talk loaded conditionally

```

117 \bool_if:NT \l__mmzx_opt_tag_bool
118 {
119   <!debug> \RequirePackage{\g__mmzx_name_str -tag}
120   <debug> \RequirePackage{\g__mmzx_name_str -tag-debug}
121   \hook_gput_code:nnn {package/forest/after}{.}
122   {
123     \hook_gput_code:nnn {begindocument/before}{.}
124     {
125       \IfPackageLoadedF {forest-lib-ext.tagging}
126       {
127         \IfPackageLoadedF {forest-lib-ext.tagging-debug}
128         {
129           \msg_warning:nnnnnn {memoize-ext}{unsupported}{forest}
130           {forest-lib-ext.tagging.sty}{forest-ext}
131           {forest trees will not be correctly tagged and may cause fatal
132            compilation errors.}
133         }
134       }
135     }
136   }
137 }

```

memoize-ext-expl3[-debug]

```

138 \bool_if:NT \l__mmzx_opt_expl_bool
139 {
140   <!debug> \RequirePackage{\g__mmzx_name_str -expl3}
141   <debug> \RequirePackage{\g__mmzx_name_str -expl3-debug}
142 }

```

memoize-ext-l3draw[-debug]

```

143 \hook_gput_code:nnn {package/l3draw/after}{.}
144 {
145   \bool_if:NT \l__mmzx_opt_draw_bool
146   {
147     <!debug> \RequirePackage {\g__mmzx_name_str -l3draw}
148     <debug> \__mmzx_debug:n {Loading memoize-ext-l3draw-debug.}
149     <debug> \RequirePackage {\g__mmzx_name_str -l3draw-debug}
150   }
151 }

```

memoize-ext-talk[-debug]

```
152 \hook_gput_code:nnn {class/ltx-talk/after} {.  
153 {  
154 \bool_if:NT \l__mmzx_opt_talk_bool  
155 {  
156 <!debug> \RequirePackage{memoize-ext-talk}  
157 <debug> \__mmzx_debug:n {Loading memoize-ext-talk-debug.}  
158 <debug> \RequirePackage{memoize-ext-talk-debug}  
159 }  
160 }
```

NaCl | halen | salt

```
161 \toksapp\mmzSalt{  
162 Tagging status: \tag_if_active_p:  
163 }
```

messages

```
164 \msg_new:nnnn {memoize-ext}{unsupported}  
165 {  
166 \msg_warning_text:n {memoize-ext}:  
167 Non-existent or inappropriate version of #2 from #3 \msg_line_context:.  
168 #4  
169 } {  
170 memoize-ext#1 requires an appropriate version of #2 from #3.  
171 }
```

</sty>

# memoize-ext-expl3

Clea F. Rees

11782 2026-01-19

## Abstract

Provides memoize-ext-expl3 and memoize-ext-expl3-common. Part of memoize-ext.

## Contents

```
<@@=mmzx> <*common>
```

```
172 \GetIdInfo $Id: memoize-ext-expl3.dtx 11782 2026-03-17 14:49:00Z cfrees $ {Extensio
173 <!debug> \ProvidesExplPackage{\ExplFileName-common}{\ExplFileDate}{v0.0 %
174 <!debug> \ExplFileVersion}{\ExplFileDescription}
175 <debug> \ProvidesExplPackage{\ExplFileName-common-debug}{\ExplFileDate}{v0.0 %
176 <debug> \ExplFileVersion}{\ExplFileDescription}
177 %
178 <debug> \disable@package@load {memoize-ext-expl3-common-debug}
179 <debug> \disable@package@load {memoize-ext-expl3-common}
180 { Only one of memoize-ext-expl3-common and memoize-ext-expl3-common-debug
181 should be loaded.
182 Since
183 <!debug> memoize-ext-expl3-common
184 <debug> memoize-ext-expl3-common-debug
185 has been loaded,I will ignore your request for
186 <debug> memoize-ext-expl3-common
187 <!debug> memoize-ext-expl3-common-debug
188 .}

189 <!debug> \RequirePackage{memoize-ext}
190 <debug> \RequirePackage{memoize-ext-debug}
191 %
```

We don't want inconsistent names in hooks.

```
192 \SetDefaultHookLabel{memoize-ext}
```

mmzx\_replicating\_bool (*var.*) Internal variable to track whether currently replicating.

```
193 \bool_new:N \l__mmzx_replicating_bool
194 \bool_set_false:N \l__mmzx_replicating_bool
```

```

mmzx_if_replicating:TF (fn.)
mmzx_if_replicating_p: (fn.)
195 \prg_new_conditional:Npnn \_mmzx_if_replicating: {p,T,TF,F}
196 {
197   \if_bool:N \l__mmzx_replicating_bool
198   <debug> \_mmzx_debug:n {Replicating true.}
199   \prg_return_true:
200   \else:
201   <debug> \_mmzx_debug:n {Replicating false.}
202   \prg_return_false:
203   \fi:
204 }

```

`\AdviceRunIfNotReplicating` Run condition.

```

205 \cs_new:Npn \AdviceRunIfNotReplicating
206 {
207   \_mmzx_if_replicating:F
208   {
209   <debug> \_mmzx_debug:n {Not replicating,so proceeding.}
210   \ifmemoizing \AdviceRuntrue \fi
211   }
212 }

```

`if not replicating` (*pgfkey*) Style.

```

213 \mmzset{
214   auto/run if not replicating/.style = {
215     run conditions={\AdviceRunIfNotReplicating},
216   },
217 }

```

Constants

```

218 \cctab_const:Nn \c__mmzx_expl_at_cctab {
219   \cctab_select:N \c_code_cctab
220   \makeatletter
221 }
222 \cctab_const:Nn \c__mmzx_nexpl_at_cctab {
223   \cctab_select:N \c_code_cctab
224   \makeatletter
225   \int_set:Nn \tex_endlinechar:D { 13 }
226   \char_set_catcode_space:n { 9 }
227   \char_set_catcode_space:n { 32 }
228   \char_set_catcode_active:n { 126 } % tilde
229 }

```

expl3, memoizing cōd ynddo

hooks instead of [TeX SE: David Carlisle](#)

`\l__mmzx_expl_bool` (*var.*) A boolean to track whether expl3 syntax is active or not. yn lle ateb | in place of [748807](#) by David Carlisle.

```

230 \bool_new:N \l__mmzx_expl_bool

```

```

_restore_ccmemo_input: (fn.) Initialise.
_saved_mmzxExplAtBegin: (fn.)
x_saved_mmzxExplAtEnd: (fn.)
231 \cs_new_protected_nopar:Npn \__mmzx_restore_ccmemo_input: {}
232 \cs_new_protected_nopar:Npn \__mmzx_saved_mmzxExplAtBegin: {}
233 \cs_new_protected_nopar:Npn \__mmzx_saved_mmzxExplAtEnd: {}

```

Auto-switching for expl3 syntax.

```

234 \hook_gput_code:nnn {begindocument/end}{.}
235 {
236   \bool_set_false:N \l__mmzx_expl_bool
237 }

238 \hook_gput_code:nnn {begindocument/end}{.}
239 {
240   <debug>   \__mmzx_debug:n {Tracking expl3 syntax changes.}
241   \bool_set_false:N \l__mmzx_expl_bool
242   \hook_gput_code:nnn {cmd/ExplSyntaxOn/before} { . }
243   {
244     \bool_if:NF \l__mmzx_expl_bool
245     {
246       \bool_set_true:N \l__mmzx_expl_bool
247       \ifmmz@direct@ccmemo@input
248         \relax
249       \else
250         \cs_set_protected_nopar:Npn \__mmzx_restore_ccmemo_input:
251         {
252           \mmz@direct@ccmemo@inputfalse
253         }
254       \fi
255       \mmz@direct@ccmemo@inputtrue
256       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtBegin: \mmzxExplAtBegin
257       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtEnd: \mmzxExplAtEnd
258       \__mmzx_expl_at_start:
259     }
260   }

```

\ExplSyntaxOn rewrites \ExplSyntaxOff, so it doesn't work to add hook code to \ExplSyntaxOff directly.

```

261 \hook_gput_code:nnn {cmd/ExplSyntaxOn/after} { . }
262 {
263   \hook_gput_code:nnn {cmd/ExplSyntaxOff/before} { . }
264   {
265     \bool_set_false:N \l__mmzx_expl_bool
266     \cs_set_eq:NN \mmzxExplAtBegin \__mmzx_saved_mmzxExplAtBegin:
267     \cs_set_eq:NN \mmzxExplAtEnd \__mmzx_saved_mmzxExplAtEnd:
268     \__mmzx_restore_ccmemo_input:
269   }
270 }
271 }

```

fns mewnl

\\_\_mmzx\_cctab\_end: (fn.) just for symmetry ...

```

272 \cs_new_eq:NN \__mmzx_cctab_end: \cctab_end:

```

```

\__mmzx_expl_at_begin: (fn.) Convenience wrappers for cat code changes.
\__mmzx_nexpl_at_begin: (fn.)
\__mmzx_expl_at_start: (fn.) 273 \cs_new_protected_nopar:Npn \__mmzx_expl_at_begin:
\__mmzx_nexpl_at_start: (fn.) 274 {
\__mmzx_cctab_stop: (fn.) 275 \cctab_begin:N \c__mmzx_expl_at_cctab
276 }
277 \cs_new_protected_nopar:Npn \__mmzx_nexpl_at_begin:
278 {
279 \cctab_begin:N \c__mmzx_nexpl_at_cctab
280 }
281 \cs_new_nopar:Npn \__mmzx_expl_at_start:
282 {
283 \cs_set_nopar:Npn \mmzxExplAtBegin {\__mmzx_expl_at_begin:}
284 \cs_set_nopar:Npn \mmzxExplAtEnd {\__mmzx_cctab_end:}
285 }
286 \cs_new_nopar:Npn \__mmzx_nexpl_at_start:
287 {
288 \cs_set_nopar:Npn \mmzxExplAtBegin {\__mmzx_nexpl_at_begin:}
289 \cs_set_nopar:Npn \mmzxExplAtEnd {\__mmzx_cctab_end:}
290 }
291 \cs_new_protected_nopar:Npn \__mmzx_cctab_stop:
292 {
293 \cs_set_nopar:Npn \mmzxExplAtBegin {relax}
294 \cs_set_nopar:Npn \mmzxExplAtEnd {relax}
295 }

```

toks memoize (cyhoeddus yn unig)

The code here is constant but the meaning changes, so what is added to the memos reflects the configuration at the time.

```

296 \appto\mmzAtBeginMemoization{
297 <debug> \__mmzx_debug:n {Appending expl begin to ccmemo at end
298 <debug> \string\mmzAtBeginMemoization.}
299 \xtoksapp\mmzCCMemo
300 {
301 \exp_not:N \csname \mmzxExplAtBegin \exp_not:N \endcsname
302 }
303 <debug> \exp_args:No \__mmzx_debug:n {\the\mmzCCMemo}
304 }
305 <debug> \cs_log:N \mmzAtBeginMemoization
306 \preto\mmzAtEndMemoization{
307 <debug> \__mmzx_debug:n {Appending expl end to ccmemo at start
308 <debug> \string\mmzAtEndMemoization.}
309 \xtoksapp\mmzCCMemo
310 {
311 \exp_not:N \csname \mmzxExplAtEnd \exp_not:N \endcsname
312 }
313 }
314 <debug> \cs_log:N \mmzAtEndMemoization

```

init

```

315 \hook_gput_code:nnn { begindocument/end } {mmzx}
316 {
317 % % % % % % \__mmzx_cctab_stop:
318 % }

```

```
</common>
```

```
<*sty>
```

```

319 \GetIdInfo $Id: memoize-ext-expl3.dtx 11782 2026-03-17 14:49:00Z cfrees $ {Extensio
320 <!debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
321 <!debug> {v0.3.4 \ExplFileVersion}{\ExplFileDescription}
322 <debug> \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
323 <debug> {v0.3.4 \ExplFileVersion}{\ExplFileDescription}
324 %
325 <!debug> \disable@package@load {memoize-ext-expl3-debug}
326 <debug> \disable@package@load {memoize-ext-expl3}
327 { Only one of memoize-ext-expl3 and memoize-ext-expl3-debug
328 should be loaded.
329 Since
330 <!debug> memoize-ext-expl3
331 <debug> memoize-ext-expl3-debug
332 has been loaded,I will ignore your request for
333 <debug> memoize-ext-expl3
334 <!debug> memoize-ext-expl3-debug
335 .}

336 <!debug> \RequirePackage{memoize-ext}
337 <debug> \RequirePackage{memoize-ext-debug}
338 <!debug> \RequirePackage{memoize-ext-expl3-common}
339 <debug> \RequirePackage{memoize-ext-expl3-common-debug}
340 %

```

We don't want inconsistent names in hooks.

```
341 \SetDefaultHookLabel{memoize-ext}
```

expl3 replicators

todo: figure out how to maintain correct grouping ...

expl\_replicate\_\_bb\_tl (*var.*) Variables

expl\_replicate\_\_ba\_tl (*var.*)

expl\_replicate\_\_tb\_tl (*var.*)

```

342 \tl_new:N \g__mmzx_expl_replicate__bb_tl
343 \tl_new:N \g__mmzx_expl_replicate__ba_tl
344 \tl_new:N \g__mmzx_expl_replicate__tb_tl
345 \tl_gput_right:NV \g__mmzx_expl_replicate__bb_tl \c_left_brace_str
346 \tl_gput_right:Nn \g__mmzx_expl_replicate__bb_tl {#}
347 \tl_gput_right:NV \g__mmzx_expl_replicate__ba_tl \c_right_brace_str
348 \tl_gput_right:Nn \g__mmzx_expl_replicate__tb_tl {#}

```

l\_replicate\_fn\_aux:nnN (*fn.*) Generic auxiliary functions for replication. The first does the actual replicating; the

\_expl\_replicate\_\_aux:n (*fn.*) second delegates details according to the argument specification.

```

349 \cs_new:Npn \__mmzx_expl_replicate_fn_aux:nnN #1#2#3
350 {
351 \cs_if_exist:cF { __mmzx_rep_#1:#2 }
352 {
353 \int_zero:N \l__mmzx_tmpa_int
354 \tl_clear:N \l__mmzx_tmpa_tl
355 \tl_clear:N \l__mmzx_tmpc_tl
356 \tl_map_function:nN {#2} \__mmzx_expl_replicate__aux:n

```

```

357 \cs_if_exist:cF { __mmzx_rep_#1:\l__mmzx_tmpc_tl }
358 {
359   \cs_gset_protected:ce {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
360   {
361     \xtoksapp\mmzCCMemo{
362       \exp_after:wN \exp_not:N \cs:w #1:#2 \cs_end: \l__mmzx_tmpa_tl
363     }
364     \exp_not:N \expandonce \exp_not:N \AdviceOriginal \l__mmzx_tmpa_tl
365     \exp_not:N \group_end:
366     \__mmzx_tmp_cctab_stop:
367   }
368 }
369 \str_if_eq:eeF {#2}{\l__mmzx_tmpc_tl}
370 { % ych!
371   \cs_new_eq:cc {__mmzx_rep_#1:#2} {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
372 }
373 }
374 \use:c { __mmzx_rep_#1:#2 }
375 }
376 \cs_new:Npn \__mmzx_expl_replicate__aux:n #1
377 {
378   \int_incr:N \l__mmzx_tmpa_int
379   \str_case:nnF { #1 }
380   {
381     {c} { \__mmzx_expl_replicate__b: }
382     {e} { \__mmzx_expl_replicate__b: }
383     {o} { \__mmzx_expl_replicate__b: }
384     {p} { }
385     {n} { \__mmzx_expl_replicate__b: }
386     {v} { \__mmzx_expl_replicate__b: }
387     {D} { }
388     {F} { \__mmzx_expl_replicate__b: }
389     {N} { \__mmzx_expl_replicate__t: }
390     {T} { \__mmzx_expl_replicate__b: }
391     {V} { \__mmzx_expl_replicate__t: }
392     {w} { \__mmzx_expl_replicate__e: }
393   }{
394     \__mmzx_expl_replicate__e:
395   }
396 }

```

`\__mmzx_expl_replicate__b:` (*fn.*) Type-specific auxiliaries. We don't need to be very specific here. We just need to distinguish argument specifiers which expect braced groups from those which expect single tokens and from those we cannot automate.

```

397 \cs_new_nopar:Npn \__mmzx_expl_replicate__b:
398 {
399   \tl_put_right:Nn \l__mmzx_tmpc_tl {n}
400   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__bb_tl
401   \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
402   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__ba_tl
403 }
404 \cs_new_nopar:Npn \__mmzx_expl_replicate__t:
405 {
406   \tl_put_right:Nn \l__mmzx_tmpc_tl {N}
407   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__tb_tl

```

```

408 \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
409 }
410 \cs_new_nopar:Npn \__mmzx_expl_replicate_e:
411 {
412 \PackageError{mmzx}{No do,sorry. Use replicate with args instead.}{}
413 }

```

`\mmzx_expl_replicate_fn:` (*fn.*) For replicating `expl3` functions.

`\__mmzx_tmp_cctab_stop:` (*fn.*)

```

\mmzx@expl@replicate@fn
414 \cs_new_eq:NN \__mmzx_tmp_cctab_stop: \__mmzx_noop:
415 \cs_new:Npn \__mmzx_expl_replicate_fn:
416 {
417 \bool_set_false:N \l__mmzx_tmpa_bool
418 \str_if_eq:eeT {\mmzxExplAtEnd} {relax}
419 {
420 \bool_set_true:N \l__mmzx_tmpa_bool
421 \__mmzx_nexpl_at_start:
422 \xtoksapp\mmzCCMemo {
423 \exp_not:N \csname \mmzxExplAtBegin \exp_not:N \endcsname
424 }
425 \cs_set:Npn \__mmzx_tmp_cctab_stop:
426 {
427 \xtoksapp\mmzCCMemo {
428 \exp_not:N \csname \mmzxExplAtEnd \exp_not:N \endcsname
429 }
430 \__mmzx_cctab_stop:
431 }
432 }{
433 \cs_set_eq:NN \__mmzx_tmp_cctab_stop: \__mmzx_noop:
434 }
435 \group_begin:
436 \bool_set_true:N \l__mmzx_replicating_bool
437 \exp_last_unbraced:Ne \__mmzx_expl_replicate_fn_aux:nnN
438 { \exp_args:NV \cs_split_function:N \AdviceReplaced }
439 }
440 \cs_new_eq:NN \mmzx@expl@replicate@fn \__mmzx_expl_replicate_fn:

```

`o/replicate expl fn` (*pgfkey*) By default we handle `\tex_savepos:D` since this commonly requires replication in `o/replicate expl var` (*pgfkey*) the kinds of environments typically subject to memoization. Another candidate is `\int_gincr:N`, but that results in a large number of additions and it is not at all clear these are generally required or desirable. Don't do this. It breaks stuff.

```

441 \mmzset{% config <<<
442 auto/replicate expl fn/.style={
443 run if not replicating,
444 outer handler=\mmzx@expl@replicate@fn,
445 },
446 }% >>>

```

</sty>

# memoize-ext-l3draw

Clea F. Rees

11782 2026-01-19

## Abstract

Support for auto-memoization of content created with l3draw (L<sup>A</sup>T<sub>E</sub>X Project 2025a). memoize-ext-l3draw is part of memoize-ext.

## Contents

<\*sty> <@@=mmzx>

```
447 \GetIdInfo $Id: memoize-ext-l3draw.dtx 11785 2026-03-17 17:32:51Z cfrees $ {Extensi
448 \!debug \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
449 {v0.3.4 \ExplFileVersion}{\ExplFileDescription}
450 \!debug \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
451 {v0.3.4 \ExplFileVersion}{\ExplFileDescription}
452 %
453 \!debug \disable@package@load {memoize-ext-l3draw-debug}
454 \!debug \disable@package@load {memoize-ext-l3draw}
455 { Only one of memoize-ext-l3draw and memoize-ext-l3draw-debug
456 should be loaded.
457 Since
458 \!debug memoize-ext-l3draw
459 \!debug memoize-ext-l3draw-debug
460 has been loaded,I will ignore your request for
461 \!debug memoize-ext-l3draw
462 \!debug memoize-ext-l3draw-debug
463 .}
464 %
465 \!debug \RequirePackage{memoize-ext}
466 \!debug \RequirePackage{memoize-ext-debug}
467 \!debug \RequirePackage{memoize-ext-expl3}
468 \!debug \RequirePackage{memoize-ext-expl3-debug}
```

l3draw

mmzx\_draw\_id\_begin\_int (*var.*) Temporary int.

```
469 \int_new:N \g__mmzx_draw_id_begin_int
```

ce\_collect\_draw\_args:w (*fn.*) The l3draw picture environment is essentially a function with a weird argument specification, so we use a custom collector.

```

470 \cs_new:Npn \__mmzx_advice_collect_draw_args:w #1 \draw_end:
471 {
472   \toks0={ #1 \draw_end: }
473   \exp_args:No \AdviceInnerHandler {\the\toks0}
474 }

```

`\AdviceCollectDrawArguments` Public wrapper.

```

475 \cs_new:Npn \AdviceCollectDrawArguments{
476   \toks0 = {}
477   \__mmzx_advice_collect_draw_args:w
478 }

```

Default configuration. This replicates changes to `\g_draw_id_int`, but does so by executing code at the start and end of *every* memoization. There must be a more efficient way to do this ....

```

479 \mmzset{%
480   gincr draw id/.style={
481     at begin memoization={
482       \gtoksapp\mmzCCMemo
483       {
484         \UseName{int_gincr:c} {g_draw_id_int}
485       }
486     },
487   },
488   auto csname={draw_begin:}{memoize,collector=\AdviceCollectDrawArguments,gincr
draw id,},
489   auto csname={__draw_record_origin:}{run if memoizing,replicate expl fn,},
490 }

```

</sty>

# memoize-ext-sockets

Clea F. Rees

11782 2026-01-19

## Abstract

memoize-ext-sockets is part of memoize-ext.

## Contents

<\*sty> <@@=mmzx>

ltsockets

socket\_assigned\_plug:n (*fn.*) Returns the name of the plug assigned to the specified socket. This ought not use a variable internal to the format's code, but there does not seem to be a public interface. So it may break, but for now it works.

```
491 \cs_new_nopar:Npn \__mmzx_socket_assigned_plug:n #1
492 { % rhybudd: fn mewno1
493   \str_use:c { l__socket_#1_plug_str }
494 }
```

</sty>

# memoize-ext-tag

Clea F. Rees

11782 2026-01-19

## Abstract

memoize-ext-tag is part of memoize-ext. It supports tagging memoized content/the memoization of tagged content.

## Contents

Uses and/or redefines the following internals, primitives and other nefarious methods:

- `\l__socket_assigned_plug:n` thing
  - If a public interface for retrieving the plug is provided at some point, I will see if I can use that. However, they do not wish it to be expandable. This is manageable, but it is very nice having an expandable function here, so I will see if I realise, remember and can do it not-too-painfully, I guess.
- `latex-lab` variables, keys etc.
  - This is fairly fragile: patching patches to make them work with patched patches . . . .
  - But I guess the `l3keys` and `pgfkeys` are public. I'm not sure about the sockets/plugs. Are these supposed to be used by external code?
  - The use of `l__tikz_tagging_alt_tl` and `l__tikz_tagging_actualext_text_tl` is definitely ungood, but I do not currently see a better way. Hopefully most people will use the `pgfkeys` interface, as that's much more natural here?
- `\tex_savepos:D`
  - This is more-or-less routine and actually documented, so no worries really here?
- `\mmzIncludeExtern`
  - Documented as internal, certainly not something to redefine, but maybe I can persuade Sašo to add the sockets I need here?

<\*sty> <@@=mmzx>

495 \GetIdInfo \$Id: memoize-ext-tag.dtx 11782 2026-03-17 14:49:00Z cfrees \$ {Extensions

```

496 <!debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
497 <!debug> {v0.3.4 \ExplFileVersion}{\ExplFileDescription}
498 <debug> \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
499 <debug> {v0.3.4 \ExplFileVersion}{\ExplFileDescription}
500 %
501 <!debug> \disable@package@load {memoize-ext-tag-debug}
502 <debug> \disable@package@load {memoize-ext-tag}
503 { Only one of memoize-ext-tag and memoize-ext-tag-debug
504 should be loaded.
505 Since
506 <!debug> memoize-ext-tag
507 <debug> memoize-ext-tag-debug
508 has been loaded,I will ignore your request for
509 <debug> memoize-ext-tag
510 <!debug> memoize-ext-tag-debug
511 .}

512 <!debug> \RequirePackage{memoize-ext}
513 <debug> \RequirePackage{memoize-ext-debug}
514 %

```

We don't want inconsistent names in hooks.

```

515 \SetDefaultHookLabel{memoize-ext}
516 %
517 \cs_generate_variant:Nn \socket_assign_plug:nn {nV}

```

```

\g__mmzx_tagpic_int (var.) Tracking & storage variables.
  \l__mmzx_toks_tl (var.)
  \l__mmzx_ok_bool (var.)
g__mmzx_plug_orig_str (var.)
  \mmzxtagtoks (toks)
518 \bool_new:N \l__mmzx_ok_bool
519 \int_new:N \g__mmzx_tagpic_int
520 \str_new:N \g__mmzx_plug_orig_str
521 \tl_new:N \l__mmzx_toks_tl
522 \newtoks\mmzxtagtoks

```

```

\include/extern/before (socket) New sockets for extern utilisation.
\include/extern/after (socket)
523 \socket_new:nn {tagsupport/memoize/include/extern/before} {3}
524 \socket_new:nn {tagsupport/memoize/include/extern/after} {0}

```

Ulrike's pgf/tikz keys don't do anything with the arguments they receive? It only seems they do because `init` passes them also to the `l3keys`?

And so we have to pass them from `tikz` to `l3keys` and back to `tikz` . . .

This creates a problem of synchronisation. It would be nice to use module-specific names to track `latex-lab` internal variables for ease of maintenance, but I don't think this is possible. If I let a new name to a token list variable, I'll just get the current value.

On the one hand, if users use `pgfkeys` to set the values for `alt` and `actualtext`, we can easily avoid `latex-lab` internals, at least. But we do that by introducing additional variables and then have the problem of synchronisation.

On the other hand, we could avoid the synchronisation problems by using `latex-lab` internals more consistently, but then even the `pgfkeys` interface will be dependent on the internal implementation<sup>1</sup>.

<sup>1</sup>I get the prohibition on internals. I really do. But there are cases where it just makes things much

Note: will need revising when the pgfkeys version of the latex-lab code gets published.

```

525 \hook_gput_code:nnn {package/tikz/after} {.}
526 {
527   \pgfqkeys{/tikz}{
528     alt/.forward to=/mmz/alt,
529     actualtext/.forward to=/mmz/actualtext,
530     artifact/.forward to=/mmz/artifact,
531     tagging-setup/.forward to=/mmz/tagging-setup,
532     /mmzx/tagging@setup/.is family,
533     alt/.belongs to family=/mmzx/tagging@setup,
534     artifact/.belongs to family=/mmzx/tagging@setup,
535     actualtext/.belongs to family=/mmzx/tagging@setup,
536     tagging-setup/.belongs to family=/mmzx/tagging@setup,
537   }
538   \mmzset{
539     alt/.code={
540       \keys_set:nn {tikz/tagging}{alt={#1}}
541       \bool_set_true:N \l__mmzx_ok_bool
542       \str_gset:Nn \g__mmzx_plug_orig_str {alt}
543       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
544       \socket_assign_plug:nn
545         {tagsupport/memoize/include/extern/before} {mmzx/alt}
546       \socket_assign_plug:nn
547         {tagsupport/memoize/include/extern/after} {mmzx/alt}
548 <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
549     },
550     actualtext/.code={
551       \keys_set:nn {tikz/tagging}{actualtext={#1}}
552       \bool_set_true:N \l__mmzx_ok_bool
553       \str_gset:Nn \g__mmzx_plug_orig_str {actualtext}
554       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
555       \socket_assign_plug:nn
556         {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
557       \socket_assign_plug:nn
558         {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
559 <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
560     },
561     artifact/.code={
562       \keys_set:nn {tikz/tagging}{artifact}
563       \bool_set_true:N \l__mmzx_ok_bool
564       \str_gset:Nn \g__mmzx_plug_orig_str {artifact}
565       \mmzxtagtoks {}
566       \socket_assign_plug:nn
567         {tagsupport/memoize/include/extern/before} {mmzx/artifact}
568       \socket_assign_plug:nn
569         {tagsupport/memoize/include/extern/after} {mmzx/artifact}
570 <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
571     },
572     tagging-setup/.is choice,
573     tagging-setup/alt/.forward to=/mmz/alt,
574     tagging-setup/actualtext/.forward to=/mmz/actualtext,
575     tagging-setup/artifact/.forward to=/mmz/artifact,
576     tagging-setup/.unknown/.code =

```

more complicated and error-prone. And sometimes it just doesn't seem worth the additional fragility that introduces, even at the cost of some additional fragility due to the use of internals.

```

577   {
578     \exp_args:Nno
579     \keys_set:nn {tikz/tagging}{\pgfkeyscurrentname={#1}}
580   },
581   alt/.belongs to family=/mmzx/tagging@setup,
582   actualtext/.belongs to family=/mmzx/tagging@setup,
583   artifact/.belongs to family=/mmzx/tagging@setup,
584   tagging-setup/.belongs to family=/mmzx/tagging@setup,
585   tagging-setup/alt/.belongs to family=/mmzx/tagging@setup,
586   tagging-setup/actualtext/.belongs to family=/mmzx/tagging@setup,
587   tagging-setup/artifact/.belongs to family=/mmzx/tagging@setup,
588 }
589 }

```

So it is possible to do without this, but it is *much* more straightforward to do with.

The basic idea is this:

- At the start of memoization, we redefine the (internal) command `\mmzIncludeExtern`.
- In its place, we use simple wrapper around a copy of the original.
- The wrapper defines a socket before and after executing the original.

This lets us wrap utilisation of the extern in an appropriate tagging structure. At least, that's the idea.

It would be nice to assign the plug here just based on whichever plugs are installed, but it is too early, while `\mmzAtEndMemoization` is too late.

```

590 \appto\mmzAtBeginMemoization{
591   \gtoksapp\mmzCCMemo{
592     \let\mmzxIncludeExternOrig\mmzIncludeExtern
593     \let\mmzIncludeExtern\mmzxIncludeExtern
594   }
595 }

```

`\_mmzx_noop:nNnnnnnnn` (*fn.*) `\mmzIncludeExtern` only seems to be defined during utilisation, so we set up a command `include_extern:nNnnnnnnn` (*fn.*) `\mmzxIncludeExternOrig` and set it to noop here, then redefine it locally when the `\mmzxIncludeExtern` extern is utilised. `\mmzIncludeExtern` is then redefined to `\mmzxIncludeExtern`, which `\mmzxIncludeExternOrig` wraps `\mmzxIncludeExternOrig` in a tagging structure<sup>2</sup>.

Note this not only uses, but redefines an internal command which the manual says users should never need to even use ....

On the other hand, this is exactly the kind of thing `memoize` does to *other* packages' internals and, indeed, to the L<sup>A</sup>T<sub>E</sub>X Project's. Which is more than can be said to defend my use of their internals ....

But does that lessen my guilt? :-)

```

596 \cs_new_protected_nopar:Npn
597   \_mmzx_noop:nNnnnnnnn #1#2#3#4#5#6#7#8#9 {}

```

<sup>2</sup>Note to self: check output of tests visually if you do not want documents to be inaccessible to that tiny group of people who rely on vision to read.

```

598 \cs_new_protected_nopar:Npn \__mmzx_include_extern:nNnnnnnnn #1#2#3#4#5#6#7#8#9
599 {
600   \int_gincr:N \g__mmzx_tagpic_int
601   <debug> \__mmzx_debug:n {Args: #1; #2; #3; #4; #5; #6; #7; #8; #9}
602   \socket_use:nnnn {tagsupport/memoize/include/extern/before}
603     {#3}{#4}{#5}
604   <debug> \__mmzx_debug:n {Executing original inclusion macro.}
605   \mmzxIncludeExternOrig {#1}#2{#3}{#4}{#5}{#6}{#7}{#8}{#9}
606   <debug> \__mmzx_debug:n {Closing tagging structures.}
607   \socket_use:n {tagsupport/memoize/include/extern/after}
608 }
609 \cs_new_eq:NN \mmzxIncludeExtern \__mmzx_include_extern:nNnnnnnnn
610 \cs_new_eq:NN \mmzxIncludeExternOrig \__mmzx_noop:nNnnnnnnn

```

extern/before mmzx/alt (*plug*) pgf/tikz addaswyd o latex-lab-testphase-tika.sty «<

```

611 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/alt}
612 {
613   <debug> \__mmzx_debug:n {Executing plug mmzx/alt in socket
614   <debug> tagsupport/memoize/include/extern/before.}
615   \mode_if_vertical:T
616   {
617     \if@inlabel
618       \mode_leave_vertical:
619     \else
620       \tag_socket_use:n {para/begin}
621     \fi
622   }
623   \tag_mc_end_push:
624   \tl_set:No \l__mmzx_toks_tl {\the\mmzxtagtoks}
625   \tag_struct_begin:n
626   {
627     tag=Figure,
628     alt=\l__mmzx_toks_tl,
629   }
630   \tag_mc_begin:n {tag=Figure}
631   \cs_new:cpe {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
632   {
633     \__mmzx_pgftikz_tag_bbox:ennn {mmzx-id\int_to_arabic:n {\g__mmzx_tagpic_int}}
634     {#1}{#2}{#3}
635   }
636   <debug> \cs_log:c {mmzx@tag@tikz@mark@pos@\int_to_arabic:n
637   <debug> {\g__mmzx_tagpic_int}}
638   \tag_struct_gput:ene
639   {\tag_get:n {struct_num}}
640   {attribute}
641   {
642     /O /Layout /BBox
643     [
644       \use:c
645       {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
646     ]
647   }
648   <debug> \__mmzx_debug:e {Recording xpos,
649   <debug> ypos of mmxc-id\int_to_arabic:n {\g__mmzx_tagpic_int}.}
650   \tex_savepos:D

```

```

651 \_mmzx_property_record_orig:ee
652   {mmzx-id\int_to_arabic:n {\g_mmzx_tagpic_int}}
653   {xpos,ypos}
654 \tex_savepos:D
655 }

```

»>

extern/after mmzx/alt (*plug*) pgf/tikz addaswyd o latex-lab-testphase-tika.sty bod yn onest, cafodd ei ddwyn o latex-lab yn hollol «<

```

656 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/alt}
657 {
658 <debug> \_mmzx_debug:n {Executing plug mmzx/alt in socket
659 <debug>   tagsupport/memoize/include/extern/after.}
660 \tag_mc_end:
661 \tag_struct_end:
662 \tag_mc_begin_pop:n {}
663 }

```

»>

before mmzx/actualtext (*plug*) «< addaswyd o latex-lab-testphase-tika.sty

after mmzx/actualtext (*plug*)

```

664 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
665 {
666 <debug> \_mmzx_debug:n {Executing plug mmzx/actualtext in socket
667 <debug>   tagsupport/memoize/include/extern/before.}
668 \keys_set:nn {tikz/tagging} {actualtext:o = {\the\mmzxtagtoks},}
669 \socket_use:n {tagsupport/tikz/picture/begin}
670 }
671 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
672 {
673 <debug> \_mmzx_debug:n {Executing plug mmzx/actualtext in socket
674 <debug>   tagsupport/memoize/include/extern/after.}
675 \socket_use:n {tagsupport/tikz/picture/end}
676 }

```

»>

/before mmzx/artifact (*plug*) «< addaswyd o latex-lab-testphase-tika.sty

n/after mmzx/artifact (*plug*)

```

677 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/artifact}
678 {
679 <debug> \_mmzx_debug:n {Executing plug mmzx/artifact in socket
680 <debug>   tagsupport/memoize/include/extern/before.}
681 \keys_set:nn {tikz/tagging} {artifact,}
682 \socket_use:n {tagsupport/tikz/picture/begin}
683 }
684 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/artifact}
685 {
686 <debug> \_mmzx_debug:n {Executing plug mmzx/artifact in socket
687 <debug>   tagsupport/memoize/include/extern/after.}
688 \socket_use:n {tagsupport/tikz/picture/end}
689 }

```

»>

`_pgftikz_tag_bbox:nmmn` (*fn.*) A memoize-friendly alternative to get the bounding box for the `alt` plug.

```

_pgftikz_tag_bbox:enmm (fn.)
690 \cs_new_nopar:Npn \__mmzx_pgftikz_tag_bbox:nmmn #1#2#3#4
691 {
692   \__mmzx_pgftikz_tag_bbox_aux:enmm
693   {
694     \mmzx_property_ref_orig:ee {#1}{xpos}
695   }
696   {
697     \mmzx_property_ref_orig:ee {#1}{ypos}
698   }
699   {#2}{#3}{#4}
700 }
701 \cs_generate_variant:Nn \__mmzx_pgftikz_tag_bbox:nmmn {enmm}

```

`ikz_tag_bbox_aux:nmmmm` (*fn.*) Auxiliary.

```

ikz_tag_bbox_aux:enmmmm (fn.)
702 \cs_new_nopar:Npn \__mmzx_pgftikz_tag_bbox_aux:nmmmm #1#2#3#4#5
703 {
704   \dim_to_decimal_in_bp:n {#1sp}
705   \c_space_tl
706   \dim_to_decimal_in_bp:n {#2sp-#5}
707   \c_space_tl
708   \dim_to_decimal_in_bp:n {#1sp+#3}
709   \c_space_tl
710   \dim_to_decimal_in_bp:n {#2sp+#4+#5}
711 }
712 \cs_generate_variant:Nn \__mmzx_pgftikz_tag_bbox_aux:nmmmm {enmmmm}

713 \hook_gput_code_with_args:nmm {cmd/tikz@picture/before} {mmzx}
714 {
715   \tag_if_active:T
716   {
717     <debug>   \__mmzx_debug:n {Executing mmzx code in hook cmd/tikz@picture/before.}
718     \socket_assign_plug:nm {tagsupport/tikz/picture/init} {mmzx}
719   }
720 }

```

»>

`ikz/picture/init mmzx` (*plug*) «< Originally, I made something much more complicated, which worked, but meh. Here the idea is that *if we are memoizing, we do not tag at all*. There’s no real downside to this: a document which includes code memoized during the current run is not usable anyway and tagging the memoized code is pointless and inefficient. (Actually, so is executing the original code for use during the current run, which is, I believe, how it works. But that is not my fault.)

Instead, we tag *only the utilisation of memos*. We don’t need to get the bounding box — `memoize` already does that. All we need do is get the position on the page during utilisation. Everything else is for free.

Note that this code uses multiple `format/latex-lab` internals. If the support for `tikz` used exclusively `pgfkeys`, this would be easily avoided: we could stick to the public interface for all but one of these usages. But because it supports also `l3keys`, I don’t see a way to avoid depending on internal variables there, too. `l3keys` does not have a `.forward_to:n` or similar. There is `.inherit:n`, but that does not work at all in the same way. Filtering

and family code (below) is copied from the code I suggested for latex-lab's TikZ support (#2004).

```

721 \socket_new_plug:nnn {tagsupport/tikz/picture/init} {mmzx}
722 {
723 <debug>    \_mmzx_debug:n {Executing plug mmzx in socket
724 <debug>    tagsupport/tikz/picture/init.}
725 \bool_set_false:N \l__mmzx_ok_bool
726 \legacy_if:nT {memoizing}
727 {
728 <debug>    \_mmzx_debug:n {Arg to tagsupport/tikz/picture/init is #1.}
729 \pgfkeyssavekeyfilterstateto\mmzx@tagging@setup@saved@filterstate
730 \pgfkeysinstallkeyfilter{/pgf/key filters/active families}{}
731 \pgfqkeysactivatesinglefamilyandfilteroptions{/mmzx/tagging@setup}{/tikz}{#1}
732 <debug>    \_mmzx_debug:e { Restoring filter state: \exp_not:V
733 <debug>    \mmzx@tagging@setup@saved@filterstate }
734 \mmzx@tagging@setup@saved@filterstate
735 \bool_if:NF \l__mmzx_ok_bool
736 {
737 <debug>    \_mmzx_debug:n {
738 <debug>    No plug set for utilisation. Trying to match latex-lab plug.
739 <debug>    }

```

If the argument to the picture set the plug, it is already in the code hash. Otherwise, we add the value of the latex-lab plug to the context, using `\mmzContextExtra` and appending globally as we are memoizing by hypothesis.

```

740 \xtoksapp\mmzContextExtra {
741   tagging plug=\_mmzx_socket_assigned_plug:n {tagsupport/tikz/picture/begin}
742 }
743 \str_case_e:nn
744 {\_mmzx_socket_assigned_plug:n {tagsupport/tikz/picture/begin}}
745 {
746   {alt} {
747     \exp_args:Ne \mmzset{
748       alt = \exp_not:V \l__tikz_tagging_alt_tl,
749     }
750 <debug>    \_mmzx_debug:n {Using alt plug.}
751   }
752   {actualtext} {
753     \exp_args:Ne \mmzset{
754       actualtext = \exp_not:V \l__tikz_tagging_actualtext_tl,
755     }
756 <debug>    \_mmzx_debug:n {Using actualtext plug.}
757   }
758   {artifact} {
759     \mmzset{artifact,}
760 <debug>    \_mmzx_debug:n {Using artifact plug.}
761   }
762   {text} {
763     \bool_set_false:N \l__mmzx_ok_bool
764 <debug>    \_mmzx_debug:e {Found unsupported
765 <debug>    text
766 <debug>    {tagsupport/tikz/picture/begin} plug.}
767 \PackageWarning{memoize-ext}{Unsupported tag config for tikz picture.
768   Please use alt,actualtext,artifact or another supported plug.
769   Note that text (the latex-lab default) is NOT supported.

```

```

770             Aborting memoization and marking code unmemoizable.
771         }
772         \mmzUnmemoizable
773     }
774 }
775 }
776 }
777 \bool_if:NT \l__mmzx_ok_bool
778 {
779     \xtoksapp\mmzCCMemo{
780         \exp_not:N \mmzxtagtoks
781         =
782         \c_left_brace_str
783         \the\mmzxtagtoks
784         \c_right_brace_str
785         \exp_not:N \AssignSocketPlug
786         \c_left_brace_str
787         tagsupport/memoize/include/extern/before
788         \c_right_brace_str
789         \c_left_brace_str
790         \__mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/before}
791         \c_right_brace_str
792         \exp_not:N \AssignSocketPlug
793         \c_left_brace_str
794         tagsupport/memoize/include/extern/after
795         \c_right_brace_str
796         \c_left_brace_str
797         \__mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/after}
798         \c_right_brace_str
799     }
800 <debug>     \__mmzx_debug:n {Disabling tagging for current tikz picture during
801 <debug>     current run.}
802     \socket_assign_plug:nn {tagsupport/tikz/picture/begin} {noop}
803     \socket_assign_plug:nn {tagsupport/tikz/picture/end} {noop}
804     \socket_assign_plug:nn {tagsupport/tikz/picture/text/begin} {noop}
805     \socket_assign_plug:nn {tagsupport/tikz/picture/text/end} {noop}
806 }
807 }

808 \hook_gput_code:nnn {cmd/mmzAbort/after} {.}
809 {
810     \legacy_if:nT {memoizing} {
811         \PackageWarning{memoize-ext}{
812             Memoization has been aborted. If something like a reference needs
813             resolving, just compile again. If the content is unmemoizable --
814             e.g. contains remember picture, a breakable tcolorbox or suchlike,
815             you must mark the content unmemoizable or the tagging will be
816             incorrect. You may do this automatically or manually. Aborted
817         }
818     }
819 }

```

»> pgf/tikz addaswyd o latex-lab-testphase-tika.sty

Hook rules to ensure our substitutes override the format's. «<

```

820 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{latex-lab-testphase-tikz}{>}{.}

```

```

821 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{tagpdf}{>}{.}
822 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{tikz}{>}{.}
823 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{latex-lab-testphase-tikz}{>}{.}
824 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{tikz}{>}{.}
825 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{tagpdf}{>}{.}
826 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {latex-lab-testphase-tikz}
827 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {tagpdf}
828 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {tikz}
829 <debug>          \hook_log:n {package/tikz/after}
830 \hook_gput_code:nnn {begindocument/end} {.}
831 {
832 <debug>          \hook_log:n {cmd/tikz@picture/before}
833 <debug>          \hook_log:n {cmd/endpgfpicture/after}

```

»>

x\_property\_ref\_orig:nn (*fn.*) «< Avoid dependency on memoize-ext-properties.

```

834 \cs_if_free:NT \mmzx_property_ref_orig:nn
835 {
836   \cs_new_eq:NN \mmzx_property_ref_orig:nn \property_ref:nn
837   \cs_generate_variant:Nn \mmzx_property_ref_orig:nn {ee}
838 }

```

»>

property\_record\_orig:nn (*fn.*) «< Avoid dependency on memoize-ext-properties.

```

839 \cs_if_free:NT \_mmzx_property_record_orig:nn
840 {
841   \cs_new_eq:NN \_mmzx_property_record_orig:nn \property_record:nn
842   \cs_generate_variant:Nn \_mmzx_property_record_orig:nn {ee}
843 }

```

»>

844 }

</sty>

# memoize-ext-talk

Clea F. Rees

11782 2026-01-19

## Abstract

memoize-ext-talk enables memoization of ltx-talk presentations (Wright 2026). The package is part of memoize-ext.

memoize-ext-talk is essentially a ‘translation’ of memoize’s support for beamer, together with modifications for differences in the way the classes implement overlays and changes to the implementation of opacity when pdfmanagement-testphase (L<sup>A</sup>T<sub>E</sub>X Project 2026) is loaded.

The package tries to avoid utilising the internals of either memoize or ltx-talk, but it was not entirely possible to do so without making the code both more fragile and unduly convoluted. See the main manual for memoize-ext for details.

The user interface is identical to that provided by memoize for beamer (Živanović 2024).

## Contents

```
<*sty> <@@=mmzx>
```

```
845 \GetIdInfo $Id: memoize-ext-talk.dtx 11782 2026-03-17 14:49:00Z cfrees $ {Extension
846 \!debug} \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
847 \!debug} {v0.3.4 \ExplFileVersion}{\ExplFileDescription}
848 \!debug} \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
849 \!debug} {v0.3.4 \ExplFileVersion}{\ExplFileDescription}
850 %
851 \!debug} \disable@package@load {memoize-ext-talk-debug}
852 \!debug} \disable@package@load {memoize-ext-talk}
853 { Only one of memoize-ext-talk and memoize-ext-talk-debug
854 should be loaded.
855 Since
856 \!debug} memoize-ext-talk
857 \!debug} memoize-ext-talk-debug
858 has been loaded,I will ignore your request for
859 \!debug} memoize-ext-talk
860 \!debug} memoize-ext-talk-debug
861 .}
```

We have to load this (I think) prior to `\documentclass`, so cannot use `tag_if_active:TF` here. We could test for `\DocumentMetadata`, but ltx-talk already requires that. But maybe I should be testing that anyway?

```
862 \!debug} \RequirePackage{memoize-ext-tag}
863 \!debug} \RequirePackage{memoize-ext-tag-debug}
```

We don't want inconsistent names in hooks.

```
864 \SetDefaultHookLabel{memoize-ext}
```

BEGIN ltx-talk addaswyd o memoize-beamer.code.tex & other memoize code

```
865 \hook_gput_code:nnn {class/ltx-talk/after}{.}
866 {
867 <debug>    \__mmzx_debug:n {Enabling ltx-talk support.}
868   \mmzset{
869     per overlay/.code={},
870     talk mode to prefix/.style={
871       prefix=\mmz@prefix@dir\mmz@prefix@name\l__mmzx_talk_mode_str -,
872     },
873   }
```

`\mmzx_talk_opacity_seq` (*var.*)

`\talk_saved_opacity_seq` (*var.*)

`\l__mmzx_talk_opacity_tl` (*var.*)

```
874 \seq_new:N \g__mmzx_talk_opacity_seq
875 \seq_new:N \g__mmzx_talk_saved_opacity_seq
876 \tl_new:N \l__mmzx_talk_opacity_tl
```

`\g__mmzx_talk_frame_int`

`\g__mmzx_talk_slide_int`

`\g__mmzx_talk_pauses_int`

`\l__mmzx_talk_mode_str`

```
877
878 \cs_new_eq:NN \g__mmzx_talk_frame_int \c@frame
879 \cs_new_eq:NN \g__mmzx_talk_slide_int \c@slide
880 \cs_new_eq:NN \g__mmzx_talk_pauses_int \c@pauses
881 \cs_new_eq:NN \l__mmzx_talk_mode_str \l__talk_mode_str
```

`\opacity_select:V` (*fn.*) ltx-talk does this too late (at least, I get an error)

```
882 \cs_generate_variant:Nn \opacity_select:n {V}
```

Initialise sequence.

```
883 \seq_gpush:Nn \g__mmzx_talk_opacity_seq {1}
```

`\mmzx_talk_opacity_save:` (*fn.*)

```
884 \cs_new_protected_nopar:Npn \__mmzx_talk_opacity_save:
885 {
886   \seq_gset_eq:NN \g__mmzx_talk_opacity_saved_seq \g__mmzx_talk_opacity_seq
887   \seq_get:NN \g__mmzx_talk_opacity_seq \l__mmzx_talk_opacity_tl
888   \exp_args:NV \tl_if_eq:NNTF \l__mmzx_talk_opacity_tl \q_no_value
889   {
890     \seq_gpush:Nn \g__mmzx_talk_opacity_saved_seq {1}
891     \opacity_select:n {1}
892   } {
893     \seq_gpush:NV \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
894     \opacity_select:V \l__mmzx_talk_opacity_tl
895   }
896 }
```

\_talk\_opacity\_restore: (fn.)

```

897 \cs_new_protected_nopar:Npn \__mmzx_talk_opacity_restore:
898 {
899   \seq_gpop:NN \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
900   \opacity_select:V \l__mmzx_talk_opacity_tl
901   \seq_gset_eq:NN \g__mmzx_talk_opacity_seq \g__mmzx_talk_opacity_saved_seq
902 }

```

\mmzSingleExternDriver Redefine driver Even if this does not have an internal name, the redefinition uses internals in spades ... We could define a new one & I guess that's what should be done ...

```

903 \long\def\mmzSingleExternDriver#1{
904   \xtoksapp\mmzCCMemo{\mmz@maybe@quitvmode}
905   \setbox\mmz@box\mmz@capture{
906     \__mmzx_talk_opacity_save:
907     #1
908     \__mmzx_talk_opacity_restore:
909   }
910   \mmzExternalizeBox\mmz@box\mmz@temptoks
911   \xtoksapp\mmzCCMemo{\the\mmz@temptoks}
912   \mmz@maybe@quitvmode\box\mmz@box
913 }

```

The code below is iffy, I guess, since the begin/end thing here is rather illusory ...

```

914 \hook_gset_rule:nnnn {begindocument} {ltx-talk} {<} {.}
915 \hook_gput_code_with_args:nnn {cmd/opacity_select:n/before} {.}
916 {
917   \seq_gpush:Nn \g__mmzx_talk_opacity_seq {#1}
918 }
919 \hook_gput_code:nnn {cmd/opacity_end:/after}{.}
920 {
921 <debug>   \__mmzx_debug:n{Opacity after}
922   \seq_gpop:NN \g__mmzx_talk_opacity_seq \l_tmpa_tl
923 <debug>   \seq_log:N \g__mmzx_talk_opacity_seq
924 }
925 \mmzset{
926   at begin memoization={
927     \socket_use:n {mmzx/talk/memoization/begin}
928   },
929   at end memoization={
930     \socket_use:n {mmzx/talk/memoization/end}
931   },
932   per overlay/.style={
933     /mmz/context={
934       overlay=\csname theslide\endcsname,
935       pauses=\ifmemoizing
936         \mmzxTalkPauses
937       \else
938         \expandafter\the\csname c@pauses\endcsname
939       \fi
940     },
941     /utils/exec={
942       \str_if_eq:eeF {peroverlay}
943       {\__mmzx_socket_assigned_plug:n {mmzx/talk/memoization/begin}}

```

```

944     {
945         \socket_assign_plug:nn {mmzx/talk/memoization/begin}{peroverlay}
946         \socket_assign_plug:nn {mmzx/talk/memoization/end}{peroverlay}

```

This is required to allow per overlay to be used in the options for tikzpictures etc.

```

947         \legacy_if:nT {memoizing} {
948             \socket_use:n {mmzx/talk/memoization/begin}
949         }
950     }
951 },

```

Is resetting the style meant to prevent duplication in memos etc.? Because it obviously won't ...?

```

952     /mmz/per overlay/.code={},
953 },
954 }

```

k/memoization/begin (*socket*) Sockets.

talk/memoization/end (*socket*)

```

955 \socket_new:nn {mmzx/talk/memoization/begin}{0}
956 \socket_new:nn {mmzx/talk/memoization/end}{0}

```

tion/begin peroverlay (*plug*) Plugs.

memoization/end peroverlay (*plug*)

```

957 \socket_new_plug:nnn {mmzx/talk/memoization/begin}{peroverlay}
958 {
959 <debug> \_mmzx_debug:n {Executing plug peroverlay in socket
960 <debug> mmzx/talk/memoization/begin.}
961 \xdef\mmzxTalkPauses{
962     \int_to_arabic:n {\g__mmzx_talk_pauses_int}
963 }
964 \xtoksapp\mmzCMemo{
965     \noexpand\mmzxSetTalkOverlays{\mmzxTalkPauses}{
966     \int_to_arabic:n {\g__mmzx_talk_slide_int}
967 }
968 }
969 \gtoksapp\mmzCCMemo{
970     \only<all:\mmzxTalkOverlays>{}
971 }
972 \seq_get:NNTF \g__mmzx_talk_opacity_seq \l__mmzx_opacity_tl
973 {
974     \fp_gset:NV \l__mmzx_tmpa_fp \l__mmzx_opacity_tl
975 } {
976     \fp_gset:Nn \l__mmzx_tmpa_fp {1}
977 }
978 \xtoksapp\mmzContextExtra{
979     opacity=\fp_to_decimal:N \l__mmzx_tmpa_fp
980 }
981 }
982 \socket_new_plug:nnn {mmzx/talk/memoization/end}{peroverlay}
983 {
984 <debug> \_mmzx_debug:n {Executing plug peroverlay in socket
985 <debug> mmzx/talk/memoization/end.}
986 \xtoksapp\mmzCCMemo{
987     \exp_not:N \setcounter{pauses}

```

```

988     {\int_to_arabic:n {\g__mmzx_talk_pauses_int}}
989   }
990 }

```

`\mmzxSetTalkOverlays` Note the addition of code to set `g__talk_slide_continue_bool`. This isn't necessary for beamer and is not 'allowed' for ltx-talk, but is necessary for frames with overlay specifications in memoized content.

```

991 \cs_new_nopar:Npn \mmzxSetTalkOverlays#1#2{
992 <debug>   \_mmzx_debug:e {Executing \cs_to_str:N \mmzxSetTalkOverlays}
993   \int_compare:nNnTF {\g__mmzx_talk_pauses_int} = {#1}
994   {
995     \gdef\mmzxTalkOverlays{#2}
996     \int_compare:nNnTF {\g__mmzx_talk_slide_int} < {#2}
997     {
998       \bool_set_true:N \l__mmzx_tmpa_bool
999     }{
1000     \bool_set_false:N \l__mmzx_tmpa_bool
1001   }
1002 }{
1003   \bool_set_true:N \l__mmzx_tmpa_bool
1004 }
1005 \bool_if:NT \l__mmzx_tmpa_bool
1006 {
1007   \bool_gset_true:N \g__talk_slide_continue_bool
1008   \appto\mmzAtBeginMemoization{
1009     \gtoksapp\mmzCMemo{\mmzxSetTalkOverlays{#1}{#2}}
1010   }
1011 }
1012 }

1013 }

</sty>

```

## References

- L<sup>A</sup>T<sub>E</sub>X Project (2025a). *The l3draw Package: Core Drawing Support*. 2025-10-09. 9th Oct. 2025. CTAN: [l3experimental](#).  
 — (2025b). *The latex-lab-tikz Package: Support for the Tagging of TikZ Pictures*. v0.80d. 27th Sept. 2025. CTAN: [latex-lab](#).  
 — (2026). *The L<sup>A</sup>T<sub>E</sub>X PDF Management Bundle*. 0.96y. 23rd Jan. 2026. CTAN: [pdfmanagement-testphase](#).  
 Rees, Clea F. (2026). *forest-ext: A Collection of forest Libraries*. 0.2. 20th Feb. 2026. CTAN: [forest-ext](#).  
 Wright, Joseph (2026). *ltx-talk: A Class for Typesetting Presentations*. 0.4.4. 10th Feb. 2026. CTAN: [ltx-talk](#).  
 Živanović, Sašo (2017). *Forest: A PGF/TikZ-Based Package for Drawing Linguistic Trees*. 2.1.5. 14th July 2017. CTAN: [forest](#).  
 — (2024). *Memoize*. 1.4.1. 2nd Dec. 2024. CTAN: [memoize](#).

## Change History

v0.2	Default to OK. . . . .	29
General: See <code>init</code> . . . . .	23	
<code>tagsupport/tikz/picture/init_mmzx</code> : Avoid processing non-tagging keys too early and too often. Save and restore existing filter state. . . . .	28	
v0.3		
General: Add tagging status to <code>salt</code> . . . . .	11	
Correct and update usage details for <code>ltx-talk</code> . . . . .	3	
Load <code>memoize-ext-tag/memoize-ext-tag-debug</code> and hope <code>\DocumentMetadata</code> is sufficient in case tagging is off. . . . .	32	
<code>mmzx/talk/memoization/end</code> : Use sockets to try to keep memos cleaner. . . . .	35	
<code>mmzx/talk/memoization/end_peroverlay</code> : What is a socket without a plug? . . . . .	35	
v0.3.1		
<code>tagsupport/tikz/picture/init_mmzx</code> :		
v0.3.2	General: Fix <code>\toksapp</code> and friends courtesy Max Chernoff. . . . .	9
Modified <code>\toksapp</code> means all engines can use the same code here. . . . .	15	
v0.3.3	<code>\__mmzx_noop:n</code> : Need this defined earlier. I'm starting to understand why libraries are a thing. . . . .	10
<code>\mmzx@expl@replicate@fn</code> : Switch cat codes if necessary. . . . .	18	
v0.3.4	General: Hopefully slightly saner code for incrementing <code>l3draw</code> id. . . . .	20
Replicate changes to <code>l3draw</code> 's id. . . . .	20	
<code>\g__mmzx_draw_id_begin_int</code> : Save and compare <code>l3draw</code> id. . . . .	19	

## Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
<code>\@ifl@t@r</code> . . . . .	5, 65
<code>\@ifundefined</code> . . . . .	2
<code>\__mmzx_advice_collect_draw_args:w</code> (fn.) . . . . .	<u>470</u> , 477
<code>\__mmzx_cctab_end</code> : (fn.) . . . . .	<u>272</u>
<code>\__mmzx_cctab_stop</code> : (fn.) . . . . .	<u>273</u> , 317, 430
<code>\__mmzx_debug:N</code> . . . . .	110
<code>\__mmzx_debug:e</code> . . . . .	112, 648, 732, 764, 992
<code>\__mmzx_debug:n</code> . . . . .	105, 109, 148, 157, 198, 201, 209, 240, 297, 303, 307, 601, 604, 606, 613, 658, 666, 673, 679, 686, 717, 723, 728, 737, 750, 756, 760, 800, 867, 921, 959, 984
<code>\__mmzx_expl_at_begin</code> : (fn.) . . . . .	<u>273</u>



\disable@package@load ..... 25, 26, 178,  
179, 325, 326, 453, 454, 501, 502, 851, 852  
\draw\_end: ..... 470, 472

## E

\else ..... 249, 619, 937  
\else: ..... 200  
\endcsname ..... 301, 311, 423, 428, 934, 938  
\endinput ..... 13  
\etoksapp ..... 86  
\exp\_after:wN ..... 362  
\exp\_args:Ne ..... 543, 554, 747, 753  
\exp\_args:Nno ..... 578  
\exp\_args:No ..... 303, 473, 548, 559, 570  
\exp\_args:NV ..... 112, 438, 888  
\exp\_last\_unbraced:Ne ..... 437  
\exp\_not:N ..... 301, 311,  
362, 364, 365, 423, 428, 780, 785, 792, 987  
\exp\_not:n ..... 112, 548, 559, 570  
\exp\_not:V ..... 732, 748, 754  
\expandafter ..... 90, 938  
\ExpandArgs ..... 76  
\expandonce ..... 364  
expl3 (opt.) ..... 3

expl3 functions:

\\_\_mmzx\_advice\_collect\_draw\_args:w 470, 477  
\\_\_mmzx\_cctab\_end: ..... 272  
\\_\_mmzx\_cctab\_stop: ..... 273, 317, 430  
\\_\_mmzx\_expl\_at\_begin: ..... 273  
\\_\_mmzx\_expl\_at\_start: ..... 258, 273  
\\_\_mmzx\_expl\_replicate\_\_aux:n ..... 349  
\\_\_mmzx\_expl\_replicate\_\_b: .....  
..... 381, 382, 383, 385, 386, 388, 390, 397  
\\_\_mmzx\_expl\_replicate\_\_e: .. 392, 394, 397  
\\_\_mmzx\_expl\_replicate\_\_t: .. 389, 391, 397  
\\_\_mmzx\_expl\_replicate\_fn: ..... 414  
\\_\_mmzx\_expl\_replicate\_fn\_aux:nnN 349, 437  
\\_\_mmzx\_if\_replicating:TF ..... 195  
\\_\_mmzx\_if\_replicating\_p: ..... 195  
\\_\_mmzx\_include\_extern:nNnnnnnnn ... 596  
\\_\_mmzx\_nexpl\_at\_begin: ..... 273  
\\_\_mmzx\_nexpl\_at\_start: ..... 273, 421  
\\_\_mmzx\_noop:nNnnnnnnn ..... 596  
\\_\_mmzx\_pgftikz\_tag\_bbox:ennn .. 633, 690  
\\_\_mmzx\_pgftikz\_tag\_bbox:nnnn ..... 690  
\\_\_mmzx\_pgftikz\_tag\_bbox\_aux:eennn ...  
..... 692, 702  
\\_\_mmzx\_pgftikz\_tag\_bbox\_aux:nnnn .. 702  
\\_\_mmzx\_property\_record\_orig:nn .... 839  
\\_\_mmzx\_restore\_ccmemo\_input: 231, 250, 268  
\\_\_mmzx\_saved\_mmzxExplAtBegin: .....  
..... 231, 256, 266  
\\_\_mmzx\_saved\_mmzxExplAtEnd: 231, 257, 267

\\_\_mmzx\_socket\_assigned\_plug:n .....  
..... 491, 741, 744, 790, 797, 943  
\\_\_mmzx\_talk\_opacity\_restore: .. 897, 908  
\\_\_mmzx\_talk\_opacity\_save: ..... 884, 906  
\\_\_mmzx\_tmp\_cctab\_stop: ..... 366, 414  
\mmzx\_property\_ref\_orig:nn ..... 834  
\opacity\_select:V ..... 882, 894, 900  
expl3 variables:  
\g\_\_mmzx\_draw\_id\_begin\_int ..... 469  
\g\_\_mmzx\_expl\_replicate\_\_ba\_tl .. 342, 402  
\g\_\_mmzx\_expl\_replicate\_\_bb\_tl . 342, 400  
\g\_\_mmzx\_expl\_replicate\_\_tb\_tl .. 342, 407  
\g\_\_mmzx\_plug\_orig\_str . 518, 542, 553, 564  
\g\_\_mmzx\_tagpic\_int .....  
..... 518, 600, 631, 633, 637, 645, 649, 652  
\g\_\_mmzx\_talk\_opacity\_seq .....  
. 874, 883, 886, 887, 901, 917, 922, 923, 972  
\g\_\_mmzx\_talk\_saved\_opacity\_seq .... 874  
\l\_\_mmzx\_expl\_bool .....  
..... 230, 236, 241, 244, 246, 265  
\l\_\_mmzx\_ok\_bool .....  
..... 518, 541, 552, 563, 725, 735, 763, 777  
\l\_\_mmzx\_opt\_draw\_bool ..... 41, 145  
\l\_\_mmzx\_opt\_expl\_bool ..... 41, 138  
\l\_\_mmzx\_opt\_tag\_bool ..... 37, 56, 117  
\l\_\_mmzx\_replicating\_bool ... 193, 197, 436  
\l\_\_mmzx\_talk\_opacity\_tl .....  
..... 874, 887, 888, 893, 894, 899, 900  
\l\_\_mmzx\_toks\_tl ..... 518, 624, 628  
\ExplFileDate ..... 17, 20, 173,  
175, 320, 322, 448, 450, 496, 498, 846, 848  
\ExplFileDescription ..... 18, 20,  
174, 176, 321, 323, 449, 451, 497, 499, 847, 849  
\ExplFileName ..... 17, 19, 23, 173,  
175, 320, 322, 448, 450, 496, 498, 846, 848  
\ExplFileVersion ..... 18, 20,  
174, 176, 321, 323, 449, 451, 497, 499, 847, 849  
\ExplLoaderFileDate ..... 5

## F

\fi ..... 210, 254, 621, 939  
\fi: ..... 203  
\fmtversion ..... 65  
\fp\_gset:Nn ..... 976  
\fp\_gset:NV ..... 974  
\fp\_new:N ..... 95  
\fp\_to\_decimal:N ..... 979

## G

\g\_\_mmzx\_draw\_id\_begin\_int (var) ..... 469  
\g\_\_mmzx\_expl\_replicate\_\_ba\_tl (var) . 342, 402  
\g\_\_mmzx\_expl\_replicate\_\_bb\_tl (var) 342, 400  
\g\_\_mmzx\_expl\_replicate\_\_tb\_tl (var) . 342, 407

<code>\g_mmzx_name_str</code> .....	<code>\l_mmzx_opacity_tl</code> .....	972, 974
..... 22, 23, 119, 120, 140, 141, 147, 149	<code>\l_mmzx_opt_draw_bool (var)</code> .....	41, 145
<code>\g_mmzx_plug_orig_str (var)</code> 518, 542, 553, 564	<code>\l_mmzx_opt_expl_bool (var)</code> .....	41, 138
<code>\g_mmzx_tagpic_int (var)</code> .....	<code>\l_mmzx_opt_tag_bool (var)</code> .....	37, 56, 117
..... 518, 600, 631, 633, 637, 645, 649, 652	<code>\l_mmzx_opt_talk_bool</code> .....	58, 154
<code>\g_mmzx_talk_frame_int</code> .....	<code>\l_mmzx_replicating_bool (var)</code> .	193, 197, 436
..... 877	<code>\l_mmzx_talk_mode_str</code> .....	871, 877
<code>\g_mmzx_talk_opacity_saved_seq</code> .....	<code>\l_mmzx_talk_opacity_tl (var)</code> .....	..... 874, 887, 888, 893, 894, 899, 900
..... 886, 890, 893, 899, 901	<code>\l_mmzx_tmpa_bool</code> .....	..... 94, 417, 420, 998, 1000, 1003, 1005
<code>\g_mmzx_talk_opacity_seq (var)</code> .....	<code>\l_mmzx_tmpa_fp</code> .....	95, 974, 976, 979
..... 874, 883, 886, 887, 901, 917, 922, 923, 972	<code>\l_mmzx_tmpa_int</code> .....	96, 353, 378, 401, 408
<code>\g_mmzx_talk_pauses_int</code> .. 877, 962, 988, 993	<code>\l_mmzx_tmpa_seq</code> .....	101
<code>\g_mmzx_talk_saved_opacity_seq (var)</code> ... 874	<code>\l_mmzx_tmpa_str</code> .....	102
<code>\g_mmzx_talk_slide_int</code> .....	<code>\l_mmzx_tmpa_tl</code> .....	..... 98, 354, 362, 364, 400, 401, 402, 407, 408
..... 877, 966, 996	<code>\l_mmzx_tmpb_str</code> .....	103
<code>\g_talk_slide_continue_bool</code> .....	<code>\l_mmzx_tmpb_tl</code> .....	99
..... 1007	<code>\l_mmzx_tmpe_tl</code> .....	..... 100, 355, 357, 359, 369, 371, 399, 406
<code>\gdef</code> .....	<code>\l_mmzx_toks_tl (var)</code> .....	518, 624, 628
..... 995	<code>\l__talk_mode_str</code> .....	881
<code>\GetIdInfo</code> .....	<code>\l__tikz_tagging_actualtext_tl</code> .....	754
..... 16, 172, 319, 447, 495, 845	<code>\l__tikz_tagging_alt_tl</code> .....	748
<code>\group_begin:</code> .....	<code>\l_tmpa_tl</code> .....	922
..... 435	<code>\legacy_if:nT</code> .....	726, 810, 947
<code>\group_end:</code> .....	<code>\let</code> .....	592, 593
..... 365	<code>\long</code> .....	903
<code>\gtoksapp</code> .....		
..... 86, 482, 591, 969, 1009		
<b>H</b>		
<code>\hook_gput_code:nnn</code> .....		
..... 121, 123, 143, 152, 234, 238,		
242, 261, 263, 315, 525, 808, 830, 865, 919		
<code>\hook_gput_code_with_args:nnn</code> .....		713, 915
<code>\hook_gset_rule:nnnn</code> .....		820,
..... 821, 822, 823, 824, 825, 826, 827, 828, 914		
<code>\hook_log:n</code> .....		829, 832, 833
<b>I</b>		
<code>\if@inlabel</code> .....		617
<code>\if_bool:N</code> .....		197
<code>\IfFormatAtLeastTF</code> .....		65, 66, 73
<code>\ifmemoizing</code> .....		210, 935
<code>\ifmmz@direct@ccmemo@input</code> .....		247
<code>\IfPackageLoadedF</code> .....		125, 127
<code>\int_compare:nNnTF</code> .....		993, 996
<code>\int_gincr:N</code> .....		600
<code>\int_incr:N</code> .....		378
<code>\int_new:N</code> .....		96, 469, 519
<code>\int_set:Nn</code> .....		225
<code>\int_to_arabic:n</code> .....		401, 408,
..... 631, 633, 636, 645, 649, 652, 962, 966, 988		
<code>\int_zero:N</code> .....		353
<code>\iow_log:n</code> .....		107
<b>K</b>		
<code>\keys_define:nn</code> .....		41
<code>\keys_set:nn</code> .....		540, 551, 562, 579, 668, 681
<b>L</b>		
<code>l3draw (opt.)</code> .....		3
<code>\l_mmzx_expl_bool (var)</code> .....		..... 230, 236, 241, 244, 246, 265
<code>\l_mmzx_ok_bool (var)</code> .....		..... 518, 541, 552, 563, 725, 735, 763, 777
<b>M</b>		
<code>\makeatletter</code> .....		220, 224
<code>\MessageBreak</code> .....		10
<code>mmz/auto/replicate expl fn (pgfkey)</code> .....		441
<code>mmz/auto/replicate expl var (pgfkey)</code> .....		441
<code>\mmz@box</code> .....		905, 910, 912
<code>\mmz@capture</code> .....		905
<code>\mmz@direct@ccmemo@inputfalse</code> .....		252
<code>\mmz@direct@ccmemo@inputtrue</code> .....		255
<code>\mmz@maybe@quitvmode</code> .....		904, 912
<code>\mmz@prefix@dir</code> .....		871
<code>\mmz@prefix@name</code> .....		871
<code>\mmz@temptoks</code> .....		910, 911
<code>\mmzAtBeginMemoization</code> 296, 298, 305, 590, 1008		
<code>\mmzAtEndMemoization</code> .....		306, 308, 314
<code>\mmzCCMemo</code> 299, 303, 309, 361, 422, 427, 482,		548, 559, 570, 591, 779, 904, 911, 969, 986
<code>\mmzCMemo</code> .....		964, 1009
<code>\mmzContextExtra</code> .....		740, 978
<code>\mmzExternalizeBox</code> .....		910
<code>\mmzIncludeExtern</code> .....		592, 593
<code>\mmzSalt</code> .....		161
<code>\mmzset</code> .....		80,
..... 213, 441, 479, 538, 747, 753, 759, 868, 925		
<code>\mmzSingleExternDriver</code> .....		903



<b>S</b>	
<code>\seq_get:NN</code> .....	887
<code>\seq_get:NNTF</code> .....	972
<code>\seq_gpop:NN</code> .....	899, 922
<code>\seq_gpush:Nn</code> .....	883, 890, 917
<code>\seq_gpush:NV</code> .....	893
<code>\seq_gset_eq:NN</code> .....	886, 901
<code>\seq_log:N</code> .....	923
<code>\seq_new:N</code> .....	101, 874, 875
<code>\setbox</code> .....	905
<code>\setcounter</code> .....	987
<code>\SetDefaultHookLabel</code> .....	36, 192, 341, 515, 864
<code>\socket_assign_plug:nn</code> .....	517, 544, 546, 555, 557, 566, 568, 718, 802, 803, 804, 805, 945, 946
<code>\socket_new:nn</code> .....	523, 524, 955, 956
<code>\socket_new_plug:nnn</code> .....	611, 656, 664, 671, 677, 684, 721, 957, 982
<code>\socket_use:n</code> .....	607, 669, 675, 682, 688, 927, 930, 948
<code>\socket_use:nnnn</code> .....	602
sockets:	
<code>mmzx/talk/memoization/begin</code> .....	955
<code>mmzx/talk/memoization/end</code> .....	955
<code>tagssupport/memoize/include/extern/after</code> .....	7, 523
<code>tagssupport/memoize/include/extern/before</code> .....	7, 523
<code>\str_case:nnF</code> .....	379
<code>\str_case_e:nn</code> .....	743
<code>\str_gset:Nn</code> .....	542, 553, 564
<code>\str_gset:NV</code> .....	23
<code>\str_if_eq:eeF</code> .....	369, 942
<code>\str_if_eq:eeT</code> .....	418
<code>\str_new:N</code> .....	22, 102, 103, 520
<code>\str_use:c</code> .....	493
<code>\string</code> .....	298, 308
<code>\sys_if_engine luatex:F</code> .....	84
<b>T</b>	
<code>tag (opt.)</code> .....	3
<code>\tag_get:n</code> .....	639
<code>\tag_if_active:T</code> .....	715
<code>\tag_if_active:TF</code> .....	38
<code>\tag_if_active_p:</code> .....	162
<code>\tag_mc_begin:n</code> .....	630
<code>\tag_mc_begin_pop:n</code> .....	662
<code>\tag_mc_end:</code> .....	660
<code>\tag_mc_end_push:</code> .....	623
<code>\tag_socket_use:n</code> .....	620
<code>\tag_struct_begin:n</code> .....	625
<code>\tag_struct_end:</code> .....	661
<code>\tag_struct_gput:ene</code> .....	638
<code>tagssupport/memoize/include/extern/after</code> (socket) .....	7, 523
<code>tagssupport/memoize/include/extern/after</code> <code>mmzx/actualtext (plug)</code> .....	664
<code>tagssupport/memoize/include/extern/after</code> <code>mmzx/alt (plug)</code> .....	656
<code>tagssupport/memoize/include/extern/after</code> <code>mmzx/artifact (plug)</code> .....	677
<code>tagssupport/memoize/include/extern/before</code> (socket) .....	7, 523
<code>tagssupport/memoize/include/extern/before</code> <code>mmzx/actualtext (plug)</code> .....	664
<code>tagssupport/memoize/include/extern/before</code> <code>mmzx/alt (plug)</code> .....	611
<code>tagssupport/memoize/include/extern/before</code> <code>mmzx/artifact (plug)</code> .....	677
<code>tagssupport/tikz/picture/init mmzx (plug)</code> ..	721
<code>talk (opt.)</code> .....	3
<code>talk mode to prefix (pgfkey)</code> .....	4
<code>\tex_endlinechar:D</code> .....	225
<code>\tex_savepos:D</code> .....	650, 654
<code>\text_purify:n</code> .....	543, 554
<code>\the</code> .....	303, 473, 548, 559, 570, 624, 668, 783, 911, 938
<code>\tl_clear:N</code> .....	354, 355
<code>\tl_gput_right:Nn</code> .....	346, 348
<code>\tl_gput_right:NV</code> .....	345, 347
<code>\tl_if_eq:NNTF</code> .....	888
<code>\tl_if_head_eq_meaning:VNF</code> .....	88
<code>\tl_log:e</code> .....	548, 559, 570
<code>\tl_map_function:nN</code> .....	356
<code>\tl_new:N</code> ..	98, 99, 100, 342, 343, 344, 521, 876
<code>\tl_put_right:Ne</code> .....	401, 408
<code>\tl_put_right:Nn</code> .....	399, 406
<code>\tl_put_right:NV</code> .....	400, 402, 407
<code>\tl_set:No</code> .....	624
token registers:	
<code>\mmzxtagtoks</code> .....	518, 543, 554, 565, 624, 668, 780, 783
<code>\toks</code> .....	472, 473, 476
<code>\toksapp</code> .....	86, 161
<b>U</b>	
<code>\use:c</code> .....	374, 644
<code>\UseName</code> .....	484
<b>X</b>	
<code>\xdef</code> .....	961
<code>\xtoksapp</code> .....	86, 299, 309, 361, 422, 427, 740, 779, 904, 911, 964, 978, 986