

# The latex-lab-testphase-l3doc package\*

Ulrike Fischer, L<sup>A</sup>T<sub>E</sub>X Project

October 30, 2026

## Contents

|          |                                                     |          |
|----------|-----------------------------------------------------|----------|
| <b>1</b> | <b>Introduction</b>                                 | <b>1</b> |
| <b>2</b> | <b>implementation</b>                               | <b>1</b> |
| 2.1      | Declaration . . . . .                               | 2        |
| 2.2      | Tag and roles . . . . .                             | 2        |
| 2.3      | Variables . . . . .                                 | 3        |
| 2.4      | The <code>function</code> environment . . . . .     | 3        |
| 2.5      | The <code>syntax</code> environment . . . . .       | 4        |
| 2.6      | The <code>macro</code> environment . . . . .        | 5        |
| 2.7      | The <code>macrocode</code> environment . . . . .    | 8        |
| 2.8      | New doc elements . . . . .                          | 9        |
| 2.9      | <code>\maketitle</code> . . . . .                   | 11       |
| 2.10     | Sectioning commands and table of contents . . . . . | 11       |
| 2.11     | Support <code>fancyvrb</code> . . . . .             | 12       |

## 1 Introduction

This package provides tagging support for the l3doc class. It is work in progress. It is neither guaranteed that every element is correctly tagged nor that the output looks identical to the untagged version.

If a documentation is tagged with this code, it is important to check the log-file for warnings, the tag structure for missing parts and the output for deviations!

Feedback at <https://github.com/latex3/tagging-project> is welcome!

TODO: Index?

## 2 implementation

<sup>1</sup> `<*package>`

---

\*

## 2.1 Declaration

```

2 \ProvidesExplPackage {latex-lab-testphase-l3doc} {\ltxlabdoctaggingdate} {\ltxlabdoctaggingvers
3   { Tagging Support for the l3doc class }

```

some minimal error checking

```

4 \IfClassLoadedF{l3doc}
5   {\PackageError{latex-lab-testphase-l3doc}{This~package~requires~the~l3doc~class!}{}}

6 \IfDocumentMetadataF
7   {\PackageError{latex-lab-testphase-l3doc}{This~package~requires~\textbackslash DocumentMeta

```

## 2.2 Tag and roles

The exact tagging structure is not yet clear. Especially for verbatim and code material. So we define roles that we can exchange when needed.

```

8 \tagpdfsetup
9 {
10   role/new-tag=function/Sect,
11   role/new-tag=functionnames/Caption,
12   role/new-tag=functionname/Code,
13   role/new-tag=functionnamepart/Span,
14   role/new-tag=syntax/Sect,
15   role/new-tag=function-description/Div,
16   role/new-tag=macro/Sect,
17   role/new-tag=macronames/Caption,
18   role/new-tag=macroname/Span,
19   role/new-tag=variable/Sect,
20   role/new-tag=variablenames/Caption,
21   role/new-tag=variablename/Span,
22   role/new-tag=codelinenum/Span,
23   % role/new-tag=part/H1 %???????
24 }

25
26 \NewStructureName{doc/function}
27 \AssignStructureRole{doc/function}{function}
28 \NewStructureName{doc/functionnames}
29 \AssignStructureRole{doc/functionnames}{functionnames}
30 \NewStructureName{doc/functionname}
31 \AssignStructureRole{doc/functionname}{functionname}
32 \NewStructureName{doc/functionnamepart}
33 \AssignStructureRole{doc/functionnamepart}{functionnamepart}
34 \NewStructureName{doc/syntax}
35 \AssignStructureRole{doc/syntax}{syntax}
36 \NewStructureName{doc/function-description}
37 \AssignStructureRole{doc/function-description}{function-description}
38 \NewStructureName{doc/macro}
39 \AssignStructureRole{doc/macro}{macro}
40 \NewStructureName{doc/macronames}
41 \AssignStructureRole{doc/macronames}{macronames}
42 \NewStructureName{doc/macroname}
43 \AssignStructureRole{doc/macroname}{macroname}
44 \NewStructureName{doc/variable}
45 \AssignStructureRole{doc/variable}{variable}

```

```

46 \NewStructureName{doc/variablenames}
47 \AssignStructureRole{doc/variablenames}{variablenames}
48 \NewStructureName{doc/variablename}
49 \AssignStructureRole{doc/variablename}{variablename}
50 \NewStructureName{doc/codelinenum}
51 \AssignStructureRole{doc/codelinenum}{codelinenum}
52 \NewStructureName{doc/codeline}
53 \AssignStructureRole{doc/codeline}{codeline}
54

```

## 2.3 Variables

`\l__codedoc_current_names_struct_tl` The structure around the command names in the margin must be moved around. For this we define a variable that will hold the relevant current structure.

```
55 \tl_new:N\l__codedoc_current_names_struct_tl
```

*(End of definition for \l\_\_codedoc\_current\_names\_struct\_tl.)*

`\g__codedoc_macroname_structnum_seq` This holds the structure numbers of the structure around every command name in a macro environment. As the structures are created inside a box this is a global variable, which is cleared at the end of the outer macro environment.

```
56 \seq_new:N\g__codedoc_macroname_structnum_seq
```

*(End of definition for \g\_\_codedoc\_macroname\_structnum\_seq.)*

## 2.4 The function environment

`\__codedoc_function_typeset_start:` This command starts the typesetting and opens the structure.

```

57 \cs_set_protected:Npn \__codedoc_function_typeset_start:
58 {
59   \par \bigskip
60   \UseTaggingSocket{struct/begin}{tag=\UseStructureName{doc/function}}

```

The paragraph structure should start after the building of the macronames. TODO: check if we can simply move ...

```

61   \tagpdfparaOff
62   \noindent
63 }

```

*(End of definition for \\_\_codedoc\_function\_typeset\_start:.)*

`\__codedoc_typeset_functions:` TODO: `\toprule` and `\bottomrule` should be artifacts with pdf<sub>l</sub>atex (they are already with lua<sub>l</sub>atex).

```

64 \cs_set_protected:Npn \__codedoc_typeset_functions:
65 { %
66   \small\ttfamily
67   \__codedoc_target:
68   \Hy@MakeCurrentHref { HD. \int_use:N \c@HD@hypercount }

```

The function names are typeset as a tabular. We use the table code to add the tagging.

```

69   \UseTaggingSocket{struct-mc}{tag=\UseStructureName{doc/functionnames}}
70   {
71     \tagpdfsetup{table/tagging=div}
72     \AssignStructureRole{table/row}{\UseStructureName{doc/functionname}}

```

```

73      \AssignStructureRole{table/cell}{\UseStructureName{doc/functionnamepart}}
74      \begin{tabular} [t] { @{} l @{} >{\hspace{\tabcolsep}} r @{} }
75      \toprule
76      \__codedoc_function_extra_labels:
77      \__codedoc_names_typeset:
78      \__codedoc_typeset_dates:
79      \bottomrule
80      \end{tabular}
81    }
82    \normalfont\normalsize\par
83  }

```

(End of definition for `\__codedoc_typeset_functions:.`)

`\__codedoc_function_descr_start:w` The function environment is mainly a box. We store the structure for the syntax environment.

```

84 \cs_set_protected:Npn \__codedoc_function_descr_start:w
85 {
86   \vcoffin_set:Nnw \l__codedoc_descr_coffin { \textwidth }
87   \tagpdfpara0n
88   \tagstructbegin{tag=\UseStructureName{doc/function-description}}
89   \tl_set:N\l__codedoc_current_names_struct_tl{\tag_get:n{struct_num}}
90   \noindent \ignorespaces
91 }

```

(End of definition for `\__codedoc_function_descr_start:w`.)

`\__codedoc_function_typeset_stop:` At the end we must close the two main structures.

```

92 \cs_set_protected:Npn \__codedoc_function_typeset_stop:
93 {
94   \par
95   \UseTaggingSocket{struct/end}% function-description
96   \UseTaggingSocket{struct/end} % function
97   \dim_set:Nn \prevdepth { \coffin_dp:N \l__codedoc_descr_coffin }
98   \allowbreak
99 }

```

(End of definition for `\__codedoc_function_typeset_stop:.`)

## 2.5 The syntax environment

These environments are full of boxes that are moved around. Getting the mc-chunks right is not trivial.

TODO: check if one should change/simplify the tagging of the inner minipage.

`\__codedoc_syntax:w`

```

100 \cs_set_protected:Npn \__codedoc_syntax:w
101 {
102   \box_if_empty:NF \g__codedoc_syntax_box
103   { \msg_error:nn { l3doc } { multiple-syntax } }
104   \dim_set:Nn \l__codedoc_syntax_dim
105   {
106     \textwidth
107     \bool_if:NT \l__codedoc_long_name_bool
108     { + \marginparwidth - \l__codedoc_trial_width_dim }

```

```

109     }
110     \tag_mc_end_push:
111     \hbox_gset:Nw \g__codedoc_syntax_box
112     \tl_if_empty:NTF\l__codedoc_current_names_struct_tl
113     {
114         \UseTaggingSocket{struct/begin}{tag=\UseStructureName{doc/syntax}}
115     }
116     {

```

TODO: check if firstkid is the right thing.

```

117         \UseTaggingSocket{struct/begin}
118         {tag=\UseStructureName{doc/syntax},firstkid,parent=\l__codedoc_current_names_struct_tl}
119     }
120     \small \ttfamily
121     \tagpdfsetup{table/tagging=false}
122     \tagpdfparaOff
123     \arrayrulecolor{white}
124     \begin{tabular} { @{} p{\l__codedoc_syntax_dim} @{} }
125     \toprule
126     \begin{minipage}[t]{\l__codedoc_syntax_dim}
127     \raggedright
128     \obeyspaces
129     \obeylines
130     }

```

*(End of definition for \\_\_codedoc\_syntax:w.)*

\\_\_codedoc\_syntax\_end:

```

131 \cs_set_protected:Npn \__codedoc_syntax_end:
132 {
133     \end{minipage}
134     \end{tabular}
135     \arrayrulecolor{black}
136     \UseTaggingSocket{struct/end}
137     \hbox_gset_end:
138     \tag_mc_begin_pop:n{}
139     \bool_if:NF \l__codedoc_in_function_bool
140     {
141         \begin{quote}
142         \mode_leave_vertical:
143         \box_use_drop:N \g__codedoc_syntax_box
144         \end{quote}
145     }
146 }

```

*(End of definition for \\_\_codedoc\_syntax\_end:.)*

## 2.6 The macro environment

The macro environment is difficult: It collects various content in boxes and outputs everything at the end in \\_\_codedoc\_macro\_dump:, including the command names in the argument which are output in the margin. A macro environment can have more than one name in the argument and the environment be nested and the respective command names are then combined with the outer names. A command name can link to a **function**

environment, which mean that it isn't simple text but contains a structure. This means that one has to collect structure numbers to insert them when needed.

`\__codedoc_macro_dump:` Here we have larger changes as we need also to replace the trivlist by a displayblock. We setup a tagging socket which will create the container for the macronames structure. The argument will contain the structure number.

```

147 \NewTaggingSocket{doc/container/macronames}{1}
148 \NewTaggingSocketPlug{doc/container/macronames}{default}
149 {
150   \tagstructbegin{tag=\UseStructureName{doc/macronames}}
151     \tl_set:Nn #1 {\tag_get:n{struct_num}}
152     \seq_map_inline:Nn \g__codedoc_macroname_structnum_seq
153       {\tag_struct_use_num:n {##1}}
154     \seq_gclear:N \g__codedoc_macroname_structnum_seq
155   \tagstructend
156 }
157 \AssignTaggingSocketPlug{doc/container/macronames}{default}
158
159 \cs_set_protected:Npn \__codedoc_macro_dump:
160 {
161   \int_compare:nNnF{\l__codedoc_nested_macro_int}>{1}
162   {
163     \begin{displayblock} [tag-name=macro,begin-vspace=\MacroTopsep]

```

we add here a container for the macronames which are built later and use the structures from the list.

```

164   \UseTaggingSocket{doc/container/macronames}{\l__codedoc_current_names_struct_tl}
165 }
166 \noindent\llap
167 { \tagmccend
168   \hbox_unpack_drop:N \l__codedoc_macro_index_box
169   \vtop to \baselineskip
170   {
171     \vbox_unpack_drop:N \l__codedoc_macro_box
172     \vss
173   }
174   \hspace{\labelsep}
175   \tagmcbegin{}
176 }
177 }

```

*(End of definition for \\_\_codedoc\_macro\_dump:.)*

`\__codedoc_macro_typeset_one:nN` The `\__codedoc_macro_typeset_one:nN` command appends one macro name to the `\l__codedoc_macro_box`. We have to add a structure. If we are on the outer level we have to record the structure number. In the inner level we can use the existing structure. As the structures are used in another place, we have to push/pop the mc.

```

178 \NewTaggingSocket{doc/macroname}{2}
179 \NewTaggingSocketPlug{doc/macroname}{default}
180 {
181   \int_compare:nNnTF {\l__codedoc_nested_macro_int} = { 1 }
182   {
183     \tagstructbegin{tag=\UseStructureName{doc/macroname},stash}
184     \seq_gput_right:Nn

```

```

185         \g__codedoc_macroname_structnum_seq{\tag_get:n{struct_num}}
186     }
187     {
188         \tagstructbegin
189         {tag=\UseStructureName{doc/macroname},parent=\l__codedoc_current_names_struct_tl}
190     }
191     \tagmcbegin{}
192     #2
193     \tagmcend
194     \tagstructend
195 }
196 \AssignTaggingSocketPlug{doc/macroname}{default}
197
198 \cs_set_protected:Npn \__codedoc_macro_typeset_one:nN #1#2
199 {
200     \tag_mc_end_push:
201     \vbox_set:Nn \l__codedoc_macro_box
202     {
203         \vbox_unpack_drop:N \l__codedoc_macro_box
204         \UseTaggingSocket{doc/macroname}{}
205         {
206             \hbox { \llap { \__codedoc_print_macroname:nN {#1} #2
207                 \MacroFont \
208                 } }
209         }
210     }
211     \tag_mc_begin_pop:n{}
212     \int_incr:N \l__codedoc_macro_int
213 }

```

(End of definition for \\_\_codedoc\_macro\_typeset\_one:nN.)

\\_\_codedoc\_macro\_reset: As we have no nested lists, we need to ignore spaces explicitly

```

214 \cs_set_protected:Npn \__codedoc_macro_reset:
215 {
216     \tl_set:Nn \l__codedoc_override_module_tl { \q_no_value }
217     \ignorespaces
218 }

```

(End of definition for \\_\_codedoc\_macro\_reset:.)

\\_\_codedoc\_macro\_end:

```

219 \cs_set_protected:Npn \__codedoc_macro_end:
220 {
221     \__codedoc_macro_end_check_tested:
222     \int_compare:nNnT \l__codedoc_nested_macro_int = 1
223     {
224         \par \__codedoc_macro_end_style:n { \__codedoc_print_end_definition: }
225         \end{displayblock}

```

Instead of a displayblock we should really use a block with a standalone recipe and it should combine two envs directly following each other within a single semantic-para, but for some reason that isn't happening and each macro env acts as if it is standalone already. TODO: figure out why that is the case.

```

226     }
227 }

```

(End of definition for `\__codedoc_macro_end:`.)

## 2.7 The macrocode environment

`\legacymacrocodesetup` We want to base the environment on the new template code. This is a counterpart to the `\legacyverbatimsetup` command in the `latex-lab-block` code.

```

228 \def\legacymacrocodesetup{%
229   \macro@font
230   \blank@linefalse \def\par{\ifblank@line
231                     \leavevmode \else \fi

```

Similar to the case for `verbatim` we must group the `\@par` so that we do not lose indentation on the first line

```

232                                     \blank@linetrue{\@par}
233                                     \penalty\interlinepenalty}
234 \obeylines
235 \@noligs
236 \let\do\@makeoother \dospecials
237 \global\@newlistfalse
238 \global\@minipagefalse
239 \ifcodeline@index
240
241   \everypar{\global\advance\c@CodelineNo\@ne
242             \llap{\tagmcend\tagstructbegin{tag=\UseStructureName{doc/codelineum}}
243             \tagmcbegin{}
244             \theCodelineNo\pdfspaces
245             \tagmcend\tagstructend
246             \tagmcbegin{}}%
247             \}%
248             \check@module}%
249 \else \everypar{\check@module}%
250 \fi
251 \tagpdfsetup{para/tag=\UseStructureName{doc/codeline}}}% check tagging
252 }

```

(End of definition for `\legacymacrocodesetup`.)

`blockenv macrocode` (*inst.*)

```

253 \DeclareInstance{blockenv}{macrocode}{std}
254 {
255   name                = macrocode,
256   tag-name            = \UseStructureName{block/verbatim},
257   tag-attr-class      = ,
258   tagging-recipe      = standard,
259   inner-level-counter = ,
260   transparent-level   = true,
261   early-legacy-code   = ,
262   late-legacy-code    = ,
263   block-instance      = verbatim ,
264   inner-instance      = ,
265   final-code          = \legacymacrocodesetup ,
266   para-instance       = justify,
267   tagging-suppress-paras = true
268 }

```

macrocode (*env*.)

```

269 \RenewDocumentEnvironment{macrocode}{!0{}}
270 {\SimpleBlockEnv{macrocode}
271   {begin-penalty = \predisplaypenalty,
272    begin-vspace = \MacrocodeTopsep,
273    left-margin = \MacroIndent,
274    para-indent = 0pt,#1}
275   \@setupverbinvisiblespace
276   \init@crossref
277   \frenchspacing \vobeyspaces
278   \xmacro@code
279 }
280 {
281   \ifpm@module \endgroup \pm@modulefalse \fi
282   \everypar{}
283   \BlockEnvEnd
284   \close@crossref
285 }

```

## 2.8 New doc elements

Various documentation elements are declare with the `\NewDocElement` command. Here we patch the relevant commands to get tagged versions.

`\@doc@env` As `\@doc@env` maps over the keys in the argument is means, that there can be more `\@doc@env@` at the begin as calls at the end, so we have to move the main structure outside of `\@doc@env@` in to `\@doc@env@`

```

286 \int_new:N\l__codedoc_nested_env_int
287 \long\def\@doc@env#1#2#3{
288   \endgroup
289   \int_incr:N\l__codedoc_nested_env_int

```

we setup a new role:

```

290 \tagpdfsetup{role/new-tag=#2/Sect}
291 \int_compare:nNnF{\l__codedoc_nested_env_int}>{1}
292 {\SimpleBlockEnv{displayblock}
293   {transparent-level=true,tag-name=#2,begin-vspace=\MacroTopsep}

```

A container for the function names.

```

294 \tagstructbegin{tag=\UseStructureName{doc/macronames}}
295 \tl_set:Nx \l__codedoc_current_names_struct_tl {\tag_get:n{struct_num}}
296 \tagstructend
297 }
298 \clist_map_inline:nn {#3} { \@doc@env@{#1}{#2}{##1} }\ignorespaces
299 }

```

(End of definition for `\@doc@env`.)

`\@doc@env@`

```

300 \long\def\@doc@env@#1#2#3{%
301   \edef\saved@macroname{\string#3}%
302   \if #1%
303     \edef\saved@indexname{\expandafter\@gobble\saved@macroname}%
304     \expandafter\ifx

```

```

305         \csname Code#2Index\endcsname
306         \CodeMacroIndex
307     \else
308         \record@index@type@save
309         {\saved@indexname}{#2}%
310     \fi
311 \else
312     \let\saved@indexname\saved@macroname
313 \fi
314 \def\makelabel##1{\noindent\llap{
315     \UseTaggingSocket{inline/begin}
316     {tag=\UseStructureName{doc/macroname},parent=\l__codedoc_current_names_struct_tl}
317     ##1\hspace{\itemsep}
318     \UseTaggingSocket{inline/end}
319     }}%
320 % \int_compare:nNnTF{\l__codedoc_nested_env_int}>{1}
321 % {
322     \let\@tempa\@empty
323     \count@\macro@cnt
324     \loop\ifnum\count@>\z@
325         \edef\@tempa{\@tempa\hbox{\strut}}\advance\count@\m@ne
326     \repeat
327     \edef\makelabel##1{\noindent\llap{
328         \UseTaggingSocket{inline/begin}
329         {tag=\UseStructureName{doc/macroname},parent=\l__codedoc_current_names_struct_tl}
330         \vtop to\baselineskip{\@tempa\hbox{##1\kern\labelsep}\vss}
331         \UseTaggingSocket{inline/end}
332         }}%
333     \advance\macro@cnt\@ne
334 % }
335 % { \macro@cnt\@ne }
336 \ifdoc@noprint
337 \else
338     \edef\@tempa{%
339         \noexpand\makelabel{
340         \noexpand\doc@providetarget
341         \noexpand\strut
342         \noexpand\@nameuse{Print#2Name}{\saved@macroname}}}%
343     \@tempa
344 \fi
345 \ifdoc@noindex\else
346     \global\advance\c@CodelineNo\@ne
347     \csname SpecialMain#2Index\expandafter\endcsname
348     \expandafter{\saved@macroname}\nobreak
349     \global\advance\c@CodelineNo\m@ne
350 \fi
351 \if#1\expandafter\DoNotIndex \expandafter {\saved@macroname}\fi
352 \ignorespaces}

```

(End of definition for \@doc@env@.)

\doc@createenv We must use \BlockEnvEnd.

```

353 \def\doc@createenv#1#2#3{%
354     \@namedef{#3}{%
355         \@ifnextchar[%]

```

```

356     {\doc@env{#1}{#2}}{\doc@env{#1}{#2}[]}}%
357 \endnamedef{end#3}
358 {
359     \int_compare:nNnF{\l__codedoc_nested_env_int}>{1}{\BlockEnvEnd}
360     \int_decr:N\l__codedoc_nested_env_int
361 }%
362 %\endnamedef{end#3}{\BlockEnvEnd}%
363 }

```

(End of definition for \doc@createenv.)

We must renew the environment doc element so that it uses the new code:

```

364 \RenewDocElement[macrolike = false ,
365                 idxtype      = env. ,
366                 idxgroup     = environments ,
367                 printtype    = \textit{env.}
368                 ]{Env}{environment}

```

## 2.9 \maketitle

The doc package redefines \maketitle. So we have to reinstate the version from the title module.

```

369 \cs_if_exist:NT \__tag_patch_maketitle:
370 {
371     \cs_set_eq:NN \maketitle \__tag_patch_maketitle:
372 }

```

## 2.10 Sectioning commands and table of contents

\l@section must get the hooks:

\l@section

```

373 \cs_gset:Npn \l@section #1#2
374 {
375     \ifnum \c@tocdepth >\z@
376         \addpenalty\@secpenalty
377         \addvspace{1.0em \@plus\p@}
378         \setlength\@tempdima{2.5em} % was 1.5em
379         \begingroup
380         \parindent \z@ \rightskip \@pnumwidth
381         \parfillskip -\@pnumwidth
382         \leavevmode \bfseries
383         \advance\leftskip\@tempdima
384         \hskip -\leftskip
385         \UseHookWithArguments{contentsline/text/before}{4}
386         {\toclevel@section}{#1}{#2}{\@contentsline@destination}%
387         \csname contentsline@text@1@format\endcsname{#1}
388         \UseHookWithArguments{contentsline/text/after}{4}
389         {\toclevel@chapter}{#1}{#2}{\@contentsline@destination}%
390         \nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss
391         \UseHookWithArguments{contentsline/page/before}{4}
392         {\toclevel@section}{#1}{#2}{\@contentsline@destination}%
393         #2

```

```

394         \UseHookWithArguments{contentsline/page/after}{4}
395         {\toclevel@section}{#1}{#2}{\@contentsline@destination}%
396         }\par
397     \endgroup
398 \fi
399 }

```

*(End of definition for \l@section.)*

## 2.11 Support fancyvrb

l3doc uses fancyvrb. As the block code redefines `verbatim` at begin document, we have to overwrite that again:

```

400 \AtBeginDocument
401 {
402     \cs_gset_eq:NN \verbatim \Verbatim
403     \cs_gset_eq:NN \endverbatim \endVerbatim
404 }
405 \end{package}

```