

The latex-lab-testphase-l3doc package*

Ulrike Fischer, L^AT_EX Project

July 11, 2025

Contents

1	Introduction	1
2	implementation	2
2.1	Declaration	2
2.2	Tag and roles	2
2.3	Variables	2
2.4	The <code>function</code> enviroment	3
2.5	The <code>syntax</code> environment	4
2.6	The <code>macro</code> environment	5
2.7	The <code>macrocode</code> environment	7
2.8	New doc elements	8
2.9	<code>\maketitle</code>	10
2.10	Sectioning commands and table of contents	10
2.11	Support <code>fancyvrb</code>	11

1 Introduction

This package provides tagging support for the l3doc class. It is work in progress. It is neither garantied 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?

TODO: update key names if block key names change

*

2 implementation

1 `<*package>`

2.1 Declaration

```
2 \ProvidesExplPackage {latex-lab-testphase-l3doc} {2025-05-07} {0.80b}
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~the~PDF~management!}{} }
```

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 }
```

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.

```
25 \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.

```
26 \seq_new:N\g__codedoc_macroname_structnum_seq

(End of definition for \g__codedoc_macroname_structnum_seq.)
```

2.4 The function enviroment

`_codedoc_function_typeset_start:` This command starts the typesetting and opens the structure.

```

27 \cs_set_protected:Npn \_codedoc_function_typeset_start:
28 {
29   \par \bigskip
30   \tagstructbegin{tag=function}

```

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

```

31   \tagpdfparaOff
32   \noindent
33 }

```

(End of definition for _codedoc_function_typeset_start:.)

`_codedoc_typeset_functions:` TODO: `\toprule` and `\bottomrule` should be artifacts with pdf_latex (they are already with lua_latex).

```

34 \cs_set_protected:Npn \_codedoc_typeset_functions:
35 { %
36   \small\ttfamily
37   \_codedoc_target:
38   \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.

```

39   \tagstructbegin{tag=functionnames}
40   \tagmcbegin{}
41   \tagpdfsetup{table/tagging=div}
42   \tl_set:Nn\l__tbl_rowtag_tl {functionname}
43   \tl_set:Nn\l__tbl_celltag_tl {functionnamepart}
44   \begin{tabular} [t] { @{} l @{} >{\hspace{\tabcolsep}} r @{} }
45     \toprule
46     \_codedoc_function_extra_labels:
47     \_codedoc_names_typeset:
48     \_codedoc_typeset_dates:
49     \bottomrule
50   \end{tabular}
51   \tagmccend
52   \tagstructend
53   \normalfont\normalsize\par
54 }

```

(End of definition for _codedoc_typeset_functions:.)

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

```

55 \cs_set_protected:Npn \_codedoc_function_descr_start:w
56 {
57   \vcoffin_set:Nnw \l__codedoc_descr_coffin { \textwidth }

```

```

58     \tagpdfpara0n
59     \tagstructbegin{tag=function-description}
60     \tl_set:N\l__codedoc_current_names_struct_tl{\tag_get:n{struct_num}}
61     \noindent \ignorespaces
62 }

```

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

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

```

63 \cs_set_protected:Npn \__codedoc_function_typeset_stop:
64 {
65     \par
66     \tagstructend % function-description
67     \tagstructend % function
68     \dim_set:Nn \prevdepth { \coffin_dp:N \l__codedoc_descr_coffin }
69     \allowbreak
70 }

```

(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`

```

71 \cs_set_protected:Npn \__codedoc_syntax:w
72 {
73     \box_if_empty:NF \g__codedoc_syntax_box
74     { \msg_error:nn { l3doc } { multiple-syntax } }
75     \dim_set:Nn \l__codedoc_syntax_dim
76     {
77         \textwidth
78         \bool_if:NT \l__codedoc_long_name_bool
79         { + \marginparwidth - \l__codedoc_trial_width_dim }
80     }
81     \tag_mc_end_push:
82     \hbox_gset:Nw \g__codedoc_syntax_box
83     \tl_if_empty:NTF\l__codedoc_current_names_struct_tl
84     {
85         \tagstructbegin{tag=syntax}
86     }
87     {

```

TODO: check if firstkid is the right thing.

```

88         \tagstructbegin{tag=syntax,firstkid,parent=\l__codedoc_current_names_struct_tl}
89     }
90     \small \ttfamily
91     \tagpdfsetup{table/tagging=false}
92     \tagpdfparaOff

```

```

93     \arrayrulecolor{white}
94     \begin{tabular} { @{} p{\l__codedoc_syntax_dim} @{} }
95         \toprule
96         \begin{minipage}[t]{\l__codedoc_syntax_dim}
97             \raggedright
98             \obeyspaces
99             \obeylines
100     }

```

(End of definition for __codedoc_syntax:w.)

__codedoc_syntax_end:

```

101 \cs_set_protected:Npn \__codedoc_syntax_end:
102 {
103     \end{minipage}
104     \end{tabular}
105     \arrayrulecolor{black}
106     \tagstructend
107     \hbox_gset_end:
108     \tag_mc_begin_pop:n{}
109     \bool_if:NF \l__codedoc_in_function_bool
110     {
111         \begin{quote}
112             \mode_leave_vertical:
113             \box_use_drop:N \g__codedoc_syntax_box
114             \end{quote}
115     }
116 }

```

(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.

```

117 \cs_set_protected:Npn \__codedoc_macro_dump:
118 {
119     \int_compare:nNnF{\l__codedoc_nested_macro_int}>{1}
120     {
121         \begin{displayblock} [tag-name=macro,beginsep=\MacroTopsep]

```

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

```

122     \tagstructbegin{tag=macronames}
123     \tl_set:Nc \l__codedoc_current_names_struct_tl {\tag_get:n{struct_num}}
124     \seq_map_inline:Nn \g__codedoc_macroname_structnum_seq
125       {\tag_struct_use_num:n {##1}}
126     \seq_gclear:N \g__codedoc_macroname_structnum_seq
127     \tagstructend
128   }
129   \noindent\llap
130     { \tagmcend
131       \hbox_unpack_drop:N \l__codedoc_macro_index_box
132       \vtop to \baselineskip
133       {
134         \vbox_unpack_drop:N \l__codedoc_macro_box
135         \vss
136       }
137       \hspace{\labelsep}
138       \tagmcbegin{}
139     }
140   }

```

(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.

```

141 \cs_set_protected:Npn \__codedoc_macro_typeset_one:nN #1#2
142   {
143     \tag_mc_end_push:
144     \vbox_set:Nn \l__codedoc_macro_box
145       {
146         \vbox_unpack_drop:N \l__codedoc_macro_box
147         \int_compare:nNnTF { \l__codedoc_nested_macro_int } = { 1 }
148         {
149           \tagstructbegin{tag=macroname,stash}
150           \seq_gput_right:Nc
151             \g__codedoc_macroname_structnum_seq{\tag_get:n{struct_num}}
152         }
153         {
154           \tagstructbegin{tag=macroname,parent=\l__codedoc_current_names_struct_tl}
155         }
156         \tagmcbegin{}
157         \hbox { \llap { \__codedoc_print_macroname:nN {#1} #2
158           \MacroFont \
159         } }
160         \tagmcend
161         \tagstructend
162       }
163     \tag_mc_begin_pop:n{}
164     \int_incr:N \l__codedoc_macro_int
165   }

```

(End of definition for __codedoc_macro_typeset_one:nN.)

`__codedoc_macro_reset:` As we have no nested lists, we need to ignore spaces explicitly

```

166 \cs_set_protected:Npn \__codedoc_macro_reset:
167   {
168     \tl_set:Nn \l__codedoc_override_module_tl { \q_no_value }
169     \ignorespaces
170   }

```

(End of definition for __codedoc_macro_reset:.)

`__codedoc_macro_end:`

```

171 \cs_set_protected:Npn \__codedoc_macro_end:
172   {
173     \__codedoc_macro_end_check_tested:
174     \int_compare:nNnT \l__codedoc_nested_macro_int = 1
175       {
176         \par \__codedoc_macro_end_style:n { \__codedoc_print_end_definition: }
177         \end{displayblock}
178       }
179   }

```

(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.

```

180 \def\legacymacrocodesetup{%
181   \macro@font
182   \blank@linefalse \def\par{\ifblank@line
183                     \leavevmode \else \fi

```

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

```

184                     \blank@linetrue{\@@par}
185                     \penalty\interlinepenalty}
186   \obeylines
187   \@noligs
188   \let\do\@makeother \dospecials
189   \global\@newlistfalse
190   \global\@minipagefalse
191   \ifcodeline@index
192
193   \everypar{\global\advance\c@CodelineNo\@ne
194             \llap{\tagmcend\tagstructbegin{tag=codelinenum}
195                 \tagmcbegin{}}
196             \theCodelineNo\pdfmakespace
197             \tagmcend\tagstructend
198             \tagmcbegin{}}%
199   \ }%
200   \check@module}%

```

```

201 \else \everypar{\check@module}%
202 \fi
203 \tagtool{paratag=codeline}% check tagging
204 }

```

(End of definition for \legacymacrocodesetup.)

blockenv macrocode (*inst.*)

```

205 \DeclareInstance{blockenv}{macrocode}{display}
206 {
207   name           = macrocode,
208   tag-name       = verbatim,
209   tag-attr-class = ,
210   tagging-recipe = standard,
211   inner-level-counter = ,
212   increment-level = false,
213   setup-code     = ,
214   block-instance = verbatimblock ,
215   inner-instance = ,
216   final-code     = \legacymacrocodesetup ,
217   para-instance  = justify,
218   tagging-suppress-paras= true
219 }

```

macrocode (*env.*)

```

220 \RenewDocumentEnvironment{macrocode}{!O{}}
221 {\UseInstance{blockenv}{macrocode}
222   {beginpenalty=\predisplaypenalty,
223     begin-skip=\MacrocodeTopsep,
224     leftmargin=\MacroIndent,
225     parindent=0pt,#1}%
226   \@setupverbinvisiblespace
227   \init@crossref
228   \frenchspacing \vobeyspaces
229   \xmacro@code
230 }
231 {
232   \ifpm@module \endgroup \pm@modulefalse \fi
233   \everypar{}
234   \endblockenv
235   \close@crossref
236 }

```

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@

```

237 \long\def\@doc@env@#1#2#3{%

```


we setup a new role:

```

238 \tagpdfsetup{role/new-tag=#2/Sect}
239 \int_incr:N\l__codedoc_nested_macro_int
240 \int_compare:nNnF{\l__codedoc_nested_macro_int}>{1}
241 {\UseInstance{blockenv}{displayblock}
242 {level-increase=false,tag-name=#2,beginsep=\MacroTopsep}

```

A container for the function names.

```

243 \tagstructbegin{tag=macronames}
244 \tl_set:Nx \l__codedoc_current_names_struct_tl {\tag_get:n{struct_num}}
245 \tagstructend
246 }
247 \edef\saved@macroname{\string#3}%
248 \if #1%
249 \edef\saved@indexname{\expandafter\@gobble\saved@macroname}%
250 \expandafter\ifx
251 \csname Code#2Index\endcsname
252 \CodeMacroIndex
253 \else
254 \record@index@type@save
255 {\saved@indexname}{#2}%
256 \fi
257 \else
258 \let\saved@indexname\saved@macroname
259 \fi
260 \def\makelabel##1{\noindent\llap{
261 \tagmcent
262 \tagstructbegin{tag=macroname,parent=\l__codedoc_current_names_struct_tl}
263 \tagmcbegin{
264 ##1\hspace{\itemsep}
265 \tagmcent
266 \tagstructend
267 \tagmcbegin{
268 }}%
269 \int_compare:nNnTF{\l__codedoc_nested_macro_int}>{1}
270 {
271 \let\@tempa\@empty
272 \count@\macro@cnt
273 \loop\ifnum\count@>\z@
274 \edef\@tempa{\@tempa\hbox{\strut}}\advance\count@\m@ne
275 \repeat
276 \edef\makelabel##1{\noindent\llap{
277 \tagmcent
278 \tagstructbegin{tag=macroname,parent=\l__codedoc_current_names_struct_tl}
279 \tagmcbegin{
280 \vtop to\baselineskip{\@tempa\hbox{##1\kern\labelsep}\vss}
281 \tagmcent
282 \tagstructend
283 \tagmcbegin{
284 }}%
285 \advance\macro@cnt\@ne
286 }

```

```

287 { \macro@cnt\@ne }
288 \ifdoc@noprint
289 \else
290   \edef\@tempa{%
291     \noexpand\makelabel{
292       \noexpand\doc@providetarget
293       \noexpand\strut
294       \noexpand\@nameuse{Print#2Name}{\saved@macroname}}}%
295   \@tempa
296 \fi
297 \ifdoc@noindex\else
298   \global\advance\c@CodelineNo\@ne
299   \csname SpecialMain#2Index\expandafter\endcsname
300   \expandafter{\saved@macroname}\nobreak
301   \global\advance\c@CodelineNo\m@ne
302 \fi
303 \if#1\expandafter\DoNotIndex \expandafter {\saved@macroname}\fi
304 \ignorespaces}

```

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

`\doc@createenv` We must use `\endblockenv`.

```

305 \def\doc@createenv#1#2#3{%
306   \@namedef{#3}{%
307     \@ifnextchar[%]
308     {\doc@env{#1}{#2}}{\doc@env{#1}{#2}[]}}%
309   \@namedef{end#3}{\endblockenv}%
310 }

```

(End of definition for \doc@createenv.)

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

```

311 \RenewDocElement[macrolike = false ,
312                 idxtype   = env.   ,
313                 idxgroup  = environments ,
314                 printtype = \textit{env.}
315                 ]{Env}{environment}

```

2.9 \maketitle

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

```

316 \cs_if_exist:NT \__tag_patch_maketitle:
317 {
318   \cs_set_eq:NN \maketitle \__tag_patch_maketitle:
319 }

```

2.10 Sectioning commands and table of contents

`\l@section` must get the hooks:

`\l@section`

```

320 \cs_gset:Npn \l@section #1#2
321 {
322   \ifnum \c@tocdepth >\z@
323     \addpenalty\@secpenalty
324     \addvspace{1.0em \@plus\p@}
325     \setlength\@tempdima{2.5em} % was 1.5em
326     \begingroup
327       \parindent \z@ \rightskip \@pnumwidth
328       \parfillskip -\@pnumwidth
329       \leavevmode \bfseries
330       \advance\leftskip\@tempdima
331       \hskip -\leftskip
332       \UseHookWithArguments{contentsline/text/before}{4}
333       {\toclevel@section}{#1}{#2}{\@contentsline@destination}%
334       \csname contentsline@text@1@format\endcsname{#1}
335       \UseHookWithArguments{contentsline/text/after}{4}
336       {\toclevel@chapter}{#1}{#2}{\@contentsline@destination}%
337       \nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss
338       \UseHookWithArguments{contentsline/page/before}{4}
339       {\toclevel@section}{#1}{#2}{\@contentsline@destination}%
340       #2
341       \UseHookWithArguments{contentsline/page/after}{4}
342       {\toclevel@section}{#1}{#2}{\@contentsline@destination}%
343       }\par
344     \endgroup
345   \fi
346 }

```

(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:

```

347 \AtBeginDocument
348 {
349   \cs_gset_eq:NN \verbatim \Verbatim
350   \cs_gset_eq:NN \endverbatim \endVerbatim
351 }
352 </package>

```