The `latex-lab-testphase-l3doc` package*

Ulrike Fischer, LaTeX Project

May 31, 2025

Contents

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

⟨*package⟩

2.1 Declaration

```
\ProvidesExplPackage {latex-lab-testphase-l3doc} {2025-05-07} {0.80b}
  { Tagging Support for the l3doc class }
```

some minimal error checking

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

\IfDocumentMetadataF
 {\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.

```
\tagpdfsetup
 {
  role/new-tag=function/Sect,
  role/new-tag=functionnames/Caption,
  role/new-tag=functionname/Code,
  role/new-tag=functionnamepart/Span,
  role/new-tag=syntax/Sect,
  role/new-tag=function-description/Div,
  role/new-tag=macro/Sect,
  role/new-tag=macronames/Caption,
  role/new-tag=macroname/Span,
  role/new-tag=variable/Sect,
  role/new-tag=variablenames/Caption,
  role/new-tag=variablename/Span,
  role/new-tag=codelinenum/Span,
  role/new-tag=part/H1
 }
```

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.

```
\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.

```
\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.

```
\cs_set_protected:Npn \__codedoc_function_typeset_start:
  {
    \par \bigskip
    \tagstructbegin{tag=function}
```

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

```
    \tagpdfparaOff
    \noindent
  }
```

(*End of definition for* `\__codedoc_function_typeset_start:`.)

__codedoc_typeset_functions: TODO: `\toprule` and `\bottomrule` should be artifacts with pdflatex (they are already with lualatex).

```
\cs_set_protected:Npn \__codedoc_typeset_functions:
  { %
    \small\ttfamily
    \__codedoc_target:
    \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.

```
    \tagstructbegin{tag=functionnames}
    \tagmcbegin{}
    \tagpdfsetup{table/tagging=div}
    \tl_set:Nn\l__tbl_rowtag_tl   {functionname}
    \tl_set:Nn\l__tbl_celltag_tl   {functionnamepart}
    \begin{tabular} [t] { @{} l @{} >{\hspace{\tabcolsep}} r @{} }
      \toprule
      \__codedoc_function_extra_labels:
      \__codedoc_names_typeset:
      \__codedoc_typeset_dates:
      \bottomrule
    \end{tabular}
    \tagmcend
    \tagstructend
    \normalfont\normalsize\par
  }
```

(*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.

```
\cs_set_protected:Npn \__codedoc_function_descr_start:w
  {
    \vcoffin_set:Nnw \l__codedoc_descr_coffin { \textwidth }
     \tagpdfparaOn
```

```
    \tagstructbegin{tag=function-description}
    \tl_set:Ne\l__codedoc_current_names_struct_tl{\tag_get:n{struct_num}}
    \noindent \ignorespaces
  }
```

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

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

```
\cs_set_protected:Npn \__codedoc_function_typeset_stop:
  {
    \par
    \tagstructend % function-description
    \tagstructend % function
    \dim_set:Nn \prevdepth { \coffin_dp:N \l__codedoc_descr_coffin }
    \allowbreak
  }
```

(*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`

```
\cs_set_protected:Npn \__codedoc_syntax:w
  {
    \box_if_empty:NF \g__codedoc_syntax_box
      { \msg_error:nn { l3doc } { multiple-syntax } }
    \dim_set:Nn \l__codedoc_syntax_dim
      {
        \textwidth
        \bool_if:NT \l__codedoc_long_name_bool
          { + \marginparwidth - \l__codedoc_trial_width_dim }
      }
    \tag_mc_end_push:
    \hbox_gset:Nw \g__codedoc_syntax_box
      \tl_if_empty:NTF\l__codedoc_current_names_struct_tl
       {
         \tagstructbegin{tag=syntax}
       }
       {
```

TODO: check if firstkid is the right thing.

```
         \tagstructbegin{tag=syntax,firstkid,parent=\l__codedoc_current_names_struct_tl}
       }
      \small \ttfamily
      \tagpdfsetup{table/tagging=false}
      \tagpdfparaOff
      \arrayrulecolor{white}
```

```
      \begin{tabular} { @{} p{\l__codedoc_syntax_dim} @{} }
        \toprule
        \begin{minipage}[t]{\l__codedoc_syntax_dim}
          \raggedright
          \obeyspaces
          \obeylines
  }
```

(*End of definition for* `\__codedoc_syntax:w`.)

`\__codedoc_syntax_end:`

```
\cs_set_protected:Npn \__codedoc_syntax_end:
  {
       \end{minipage}
      \end{tabular}
      \arrayrulecolor{black}
    \tagstructend
    \hbox_gset_end:
    \tag_mc_begin_pop:n{}
    \bool_if:NF \l__codedoc_in_function_bool
      {
        \begin{quote}
          \mode_leave_vertical:
          \box_use_drop:N \g__codedoc_syntax_box
        \end{quote}
      }
  }
```

(*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.

```
\cs_set_protected:Npn \__codedoc_macro_dump:
  {
    \int_compare:nNnF{\l__codedoc_nested_macro_int}>{1}
     {
       \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.

```
       \tagstructbegin{tag=macronames}
       \tl_set:Ne \l__codedoc_current_names_struct_tl {\tag_get:n{struct_num}}
```

```
      \seq_map_inline:Nn \g__codedoc_macroname_structnum_seq
        {\tag_struct_use_num:n {##1}}
      \seq_gclear:N \g__codedoc_macroname_structnum_seq
      \tagstructend
    }
  \noindent\llap
        { \tagmcend
          \hbox_unpack_drop:N \l__codedoc_macro_index_box
          \vtop to \baselineskip
            {
              \vbox_unpack_drop:N \l__codedoc_macro_box
              \vss
            }
          \hspace{\labelsep}
          \tagmcbegin{}
        }
  }
```

(*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.

```
\cs_set_protected:Npn \__codedoc_macro_typeset_one:nN #1#2
  {
    \tag_mc_end_push:
    \vbox_set:Nn \l__codedoc_macro_box
      {
        \vbox_unpack_drop:N \l__codedoc_macro_box
        \int_compare:nNnTF { \l__codedoc_nested_macro_int } = { 1 }
         {
           \tagstructbegin{tag=macroname,stash}
            \seq_gput_right:Ne
             \g__codedoc_macroname_structnum_seq{\tag_get:n{struct_num}}
         }
         {
           \tagstructbegin{tag=macroname,parent=\l__codedoc_current_names_struct_tl}
         }
        \tagmcbegin{}
        \hbox { \llap { \__codedoc_print_macroname:nN {#1} #2
            \MacroFont \
        } }
        \tagmcend
        \tagstructend
      }
    \tag_mc_begin_pop:n{}
    \int_incr:N \l__codedoc_macro_int
  }
```

(*End of definition for* `\__codedoc_macro_typeset_one:nN`.)

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

```
\cs_set_protected:Npn \__codedoc_macro_reset:
```

```
  {
    \tl_set:Nn \l__codedoc_override_module_tl { \q_no_value }
    \ignorespaces
  }
```

(*End of definition for* `\__codedoc_macro_reset:`.)

`\__codedoc_macro_end:`

```
\cs_set_protected:Npn \__codedoc_macro_end:
  {
    \__codedoc_macro_end_check_tested:
    \int_compare:nNnT \l__codedoc_nested_macro_int = 1
      {
        \par \__codedoc_macro_end_style:n { \__codedoc_print_end_definition: }
        \end{displayblock}
      }
  }
```

(*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.

```
\def\legacymacrocodesetup{%
  \macro@font
  \blank@linefalse \def\par{\ifblank@line
                              \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

```
                              \blank@linetrue{\@@par}
                              \penalty\interlinepenalty}
  \obeylines
  \@noligs
  \let\do\@makeother \dospecials
  \global\@newlistfalse
  \global\@minipagefalse
  \ifcodeline@index

    \everypar{\global\advance\c@CodelineNo\@ne
              \llap{\tagmcend\tagstructbegin{tag=codelinenum}
              \tagmcbegin{}
              \theCodelineNo\pdffakespace
              \tagmcend\tagstructend
              \tagmcbegin{}%
              \ }%
              \check@module}%
  \else \everypar{\check@module}%
  \fi
  \tagtool{paratag=codeline}% check tagging
  }
```

(*End of definition for* `\legacymacrocodesetup`.)

`blockenv macrocode` (*inst.*)

```
\DeclareInstance{blockenv}{macrocode}{display}
{
  env-name       = macrocode,
  tag-name       = verbatim,
  tag-class      = ,
  tagging-recipe = standard,
  inner-level-counter  = ,
  level-increase = false,
  setup-code     = ,
  block-instance = verbatimblock ,
  inner-instance = ,
  final-code     = \legacymacrocodesetup ,
  para-instance  = justify,
  para-flattened= true
}
```

`macrocode` (*env.*)

```
\RenewDocumentEnvironment{macrocode}{!O{}}
  {\UseInstance{blockenv}{macrocode}
    {beginpenalty=\predisplaypenalty,
     begin-skip=\MacrocodeTopsep,
     leftmargin=\MacroIndent,
     parindent=0pt,#1}%
    \@setupverbinvisiblespace
    \init@crossref
    \frenchspacing \@vobeyspaces
    \xmacro@code
  }
  {
   \ifpm@module \endgroup \pm@modulefalse \fi
   \everypar{}
   \endblockenv
   \close@crossref
  }
```

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@`

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

we setup a new role:

```
  \tagpdfsetup{role/new-tag=#2/Sect}
  \int_incr:N\l__codedoc_nested_macro_int
  \int_compare:nNnF{\l__codedoc_nested_macro_int}>{1}
```

```
    {\UseInstance{blockenv}{displayblock}
     {level-increase=false,tag-name=#2,beginsep=\MacroTopsep}
```

A container for the function names.

```
    \tagstructbegin{tag=macronames}
    \tl_set:Ne \l__codedoc_current_names_struct_tl {\tag_get:n{struct_num}}
    \tagstructend
    }
  \edef\saved@macroname{\string#3}%
    \if #1%
      \edef\saved@indexname{\expandafter\@gobble\saved@macroname}%
      \expandafter\ifx
                  \csname Code#2Index\endcsname
                  \CodeMacroIndex
      \else
        \record@index@type@save
          {\saved@indexname}{#2}%
      \fi
    \else
      \let\saved@indexname\saved@macroname
    \fi
    \def\makelabel##1{\noindent\llap{
      \tagmcend
      \tagstructbegin{tag=macroname,parent=\l__codedoc_current_names_struct_tl}
      \tagmcbegin{}
      ##1\hspace{\itemsep}
      \tagmcend
      \tagstructend
      \tagmcbegin{}
      }}%
    \int_compare:nNnTF{\l__codedoc_nested_macro_int}>{1}
     {
      \let\@tempa\@empty
      \count@\macro@cnt
      \loop\ifnum\count@>\z@
        \edef\@tempa{\@tempa\hbox{\strut}}\advance\count@\m@ne
      \repeat
      \edef\makelabel##1{\noindent\llap{
      \tagmcend
      \tagstructbegin{tag=macroname,parent=\l__codedoc_current_names_struct_tl}
      \tagmcbegin{}
      \vtop to\baselineskip{\@tempa\hbox{##1\kern\labelsep}\vss}
      \tagmcend
      \tagstructend
      \tagmcbegin{}
      }}%
      \advance\macro@cnt\@ne
     }
     { \macro@cnt\@ne }
    \ifdoc@noprint
    \else
      \edef\@tempa{%
        \noexpand\makelabel{
```

```
        \noexpand\doc@providetarget
        \noexpand\strut
        \noexpand\@nameuse{Print#2Name}{\saved@macroname}}}%
      \@tempa
  \fi
  \ifdoc@noindex\else
    \global\advance\c@CodelineNo\@ne
    \csname SpecialMain#2Index\expandafter\endcsname
      \expandafter{\saved@macroname}\nobreak
    \global\advance\c@CodelineNo\m@ne
  \fi
  \if#1\expandafter\DoNotIndex \expandafter {\saved@macroname}\fi
  \ignorespaces}
```

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

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

```
\def\doc@createenv#1#2#3{%
  \@namedef{#3}{%
    \@ifnextchar[%]
    {\doc@env{#1}{#2}}{\doc@env{#1}{#2}[]}}%
  \@namedef{end#3}{\endblockenv}%
}
```

(*End of definition for* `\doc@createenv`.)

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

```
\RenewDocElement[macrolike = false ,
                 idxtype   = env.  ,
                 idxgroup  = environments ,
                 printtype = \textit{env.}
                ]{Env}{environment}
```

2.9 `\maketitle`

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

```
\cs_if_exist:NT \__tag_patch_maketitle:
 {
   \cs_set_eq:NN \maketitle \__tag_patch_maketitle:
 }
```

2.10 Sectioning commands and table of contents

`\l@section` must get the hooks:

`\l@section`

```
\cs_gset:Npn \l@section #1#2
  {
    \ifnum \c@tocdepth >\z@
      \addpenalty\@secpenalty
```

```
      \addvspace{1.0em \@plus\p@}
      \setlength\@tempdima{2.5em}  % was 1.5em
      \begingroup
        \parindent \z@ \rightskip \@pnumwidth
        \parfillskip -\@pnumwidth
        \leavevmode \bfseries
        \advance\leftskip\@tempdima
        \hskip -\leftskip
        \UseHookWithArguments{contentsline/text/before}{4}
           {\toclevel@section}{#1}{#2}{\@contentsline@destination}%
        \csname contentsline@text@1@format\endcsname{#1}
        \UseHookWithArguments{contentsline/text/after}{4}
           {\toclevel@chapter}{#1}{#2}{\@contentsline@destination}%
        \nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss
        \UseHookWithArguments{contentsline/page/before}{4}
        {\toclevel@section}{#1}{#2}{\@contentsline@destination}%
        #2
        \UseHookWithArguments{contentsline/page/after}{4}
        {\toclevel@section}{#1}{#2}{\@contentsline@destination}%
        }\par
      \endgroup
    \fi
  }
```

(*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:

```
\AtBeginDocument
  {
    \cs_gset_eq:NN \verbatim \Verbatim
    \cs_gset_eq:NN \endverbatim \endVerbatim
  }
```

⟨/package⟩