

The skdoc document class^{*†}

Simon Sigurdhsson [sigurdhsson@gmail.com]

Version 0.9

Abstract The skdoc class provides macros to document the functionality and implementation of L^AT_EX packages and document classes. It is loosely based on the ydoc and ltxdoc classes, but has a number of incompatible differences. The class defines a MacroCode environment which substitutes the usual docstrip method of installing packages. It has the ability to generate both documentation and code in a single run of a single file.

Contents

1	Introduction	2
2	Documentation	2
2.1	Planned improvements	2
3	Implementation	3
3.1	Require packages	3
3.2	Error messages	4
3.3	Options	4
3.4	The MacroCode environment	5
	Reading a preamble [9]	
3.5	Styling	10
	Colors [10] Fonts [10] Configuring hyperref [11]	

^{*}Available on <http://www.ctan.org/pkg/skdoc>.

[†]Development version available on <https://github.com/urdh/skdoc>.

3.6	Documentation macros	11
	Inline referencing [11] Descriptive macros [12] Implementation environment [16]	
3.7	The index	21
	Adding index entries [22] Displaying the index [25]	
3.8	The changelog	27
	Adding changes [28] Displaying the changelog [29]	
3.9	Hiding the implementation	30
3.10	Document metadata	33
	Setting metadata [33] Using metadata [35]	
3.11	General document commands	36
	The title page [36] Table of contents [38]	
3.12	Cosmetic changes	39
4	Changes	40
5	Index	40

1 Introduction

2 Documentation

2.1 Planned improvements

Planned improvements of this class include

- Proper documentation of all features, including those inherited from ydoc.
- Do things the \LaTeX way, rename the variables properly and generally conform to expl3 guidelines when applicable.
- Better color scheme.
- `\DescribeEnvironment` has to be improved. The idea is to typeset the first line as with the macros, but the environment contents and `\end` command typeset below in a `\marginnote`.

- More organized font scheme. `\cs`, `\env`, `\pkg`, `\opt`, `\thm` and `\bib` must have sensible fonts and these should be reflected in the implementation details as well as the description.
- The uses of `\cs` and friends should be recorded in the index as well.
- Restructure the code.
- Starred variants of the implementation environments have to gobble the arguments that are normally printed!
- Themes and `\BibTeX` entry types need macros in the spirit of `\Option` and friends.

3 Implementation

3.1 Require packages

We begin with loading the `scrartcl` KOMA-script class and a few packages we'll be needing.

```

1 \LoadClass[DIV7,
2           headings=big,
3           numbers=noenddot,
4           abstracton,
5           bibliography=totoc,
6           index=totoc
7           ]{scrartcl}

```

These packages are basic low-level things that we use to declare commands, work with strings and so on.

```

8 \RequirePackage{etoolbox,xstring,xparse,atbegshi,
9               ,kvoptions}

```

Now, higher-level packages we use in our definitions.

```

9 \RequirePackage{verbatim,marginnote,calc,
10               hyperref,multicol,hologo}

```

```

10 \RequirePackage[nomain,xindy,numberedsection,✓
    order=letter,
11         sanitize={description=false,sort✓
            =false}]{glossaries}

```

We also include the ydoc packages we'll be extending.

```

12 \RequirePackage{ydoc-code,ydoc-desc}

```

The rest is basically just styling.

```

13 \RequirePackage[british]{babel}
14 \RequirePackage[babel]{microtype}
15 \RequirePackage{scrpage2,PTSerif,ascii}
16 \RequirePackage[defaultsans,osfigures,scale✓
    =0.95]{opensans}

```

3.2 Error messages

Set up some error message texts for later use.

```

17 \msg_new:nnn{skdoc}{key-exists}{File~key~"#1"~✓
    already~declared!}
18 \msg_new:nnn{skdoc}{key-nexists}{File~key~"#1"~✓
    hasn't~been~declared!}
19 \msg_new:nnn{skdoc}{wrote-file}{Writing~things~✓
    to~file~"#1".}
20 \msg_new:nnn{skdoc}{read-preamble}{Reading~✓
    preamble~from~file~"#1".}

```

3.3 Options

Define the (as of version 1.0) only option, and process it.

```

21 \SetupKeyvalOptions{
22     family=skdoc,
23     prefix=skdoc@
24 }

```

```

25 \DeclareStringOption{load}[\jobname]
26 \ProcessKeyvalOptions*
    If the option was used, load the given package if it exists.
27 \IfStrEq{\skdoc@load}{\jobname}{%
28     \IfFileExists{\skdoc@load.sty}{%
29         \RequirePackage{\skdoc@load}
30     }{}
31 }

```

3.4 The MacroCode environment

We need a token list and input/output.

```

32 \tl_new:N\skdoc@temptl
33 \ior_new:N\skdoc@input
34 \iow_new:N\skdoc@output

```

\DeclareFile #1: A list of key-value parameters

#2: Filename of the file being declared

This declares a file as part of the bundle, which means we will be writing things to it.

```

35 \DeclareDocumentCommand\DeclareFile{om}{
36     \group_begin:
37     \keys_define:nn{skdoc@declarefile}{%
38         preamble .value_required:,
39         preamble .code:n = \edef\skdoc@preamble{
40             {##1},
41             key .value_required:,
42             key .code:n = \def\skdoc@key{##1}
43         }%
44     \def\skdoc@preamble{}%
45     \def\skdoc@key{#2}%
46     \IfNoValueTF{#1}{\keys_set:nn{
47         skdoc@declarefile}{#1}}

```

```

46 \tl_new:c{skdoc@output@\skdoc@key}
47 \int_if_exist:cTF{skdoc@output@\skdoc@key ✓
    @line}{
48 \msg_critical:nnx{skdoc}{key-exists}{\✓
    skdoc@key}
49 }{
50 \int_zero_new:c{skdoc@output@\skdoc@key ✓
    @line}
51 }
52 \IfStrEq{\skdoc@preamble}{\}{\}{
53 \tl_set:Nx\l_tmpa_tl{\skdoc@preamble}
54 \edef\skdoc@temp{\tl_head:N\l_tmpa_tl}
55 \def\skdoc@preamble@extra{
56 \skdoc@temp\skdoc@temp\space~This~is✓
    ~file~'#2',~generated~from~'\c_✓
    job_name_tl.tex '~(key~'\skdoc@key✓
    ').
57 }
58 }
59 \expandafter\xdef\csname skdoc@write@#2\✓
    endcsname{%
60 \noexpand\msg_log:nnn{skdoc}{wrote-file✓
    }{#2}
61 \noexpand\iow_open:Nn\noexpand\✓
    skdoc@output{#2}
62 \noexpand\IfStrEq{\skdoc@preamble}{\}{\}{
63 \noexpand\iow_now:Nx\noexpand\✓
    skdoc@output{\✓
    skdoc@preamble@extra}
64 \noexpand\iow_now:Nx\noexpand\✓
    skdoc@output{\skdoc@preamble}
65 }
66 \noexpand\iow_now:Nx\noexpand\✓
    skdoc@output{\noexpand\tl_to_str:c{✓
    skdoc@output@\skdoc@key}}
67 \noexpand\iow_close:N\noexpand\✓

```

```

        skdoc@output
68     }
69     \AfterEndDocument{\csname skdoc@write@#2\
        endcsname}
70     \group_end:
71 }

skdoc@verbatim #1: The key of the file being described
                This environment does all the actual work for MacroCode.

72 \DeclareDocumentEnvironment{skdoc@verbatim}{m}{%
73     \iftoggle{skdoc@impl}{\@bsphack}{}
74     \def\skdoc@key{#1}
75     \int_if_exist:cTF{skdoc@output@\skdoc@key ✓
        @line}{}{
76         \msg_critical:nx{skdoc}{key-nexists}{\
        skdoc@key}
77     }%
78     \marginnote{
79         \leavevmode
80         \llap{
81             \scriptsize\color{gray}
82             $\langle$\skdoc@key$\rangle$
83             \makebox[2ex]{\strut}
84         }
85     }
86     \def\verbatim@processline{%
87         \tl_gput_right:cx{skdoc@output@\
        skdoc@key}{\the\verbatim@line\iow_✓
        newline:}%
88         \int_gincr:c{skdoc@output@\skdoc@key ✓
        @line}%
89         \iftoggle{skdoc@impl}{
90             \noindent\leavevmode%
91             \hspace*{-5ex}
92             \begin{minipage}[c][1ex]{\textwidth}
93                 \makebox[4ex]{%

```

```

94         \leavevmode
95         \tiny\color{lightgray}\hfill\
          %
96         \int_use:c{skdoc@output@\
          skdoc@key @line}%
97     }%
98     \hspace*{1ex}%
99     {
100         \verbatim@font\footnotesize
101         \the\verbatim@line
102     }
103     \end{minipage}
104     \vskip-.75ex\par
105 }{}
106 }%
107 \let\do\@makeother\dospecials\catcode'\^^M\
    active%
108 \iftoggle{skdoc@impl}{
109     \frenchspacing\@vobeyspaces
110 }{}
111 \verbatim@start%
112 }{%
113     \iftoggle{skdoc@impl}{\@esphack}{}%
114 }

```

MacroCode #1: The key of the file being described

```

115 \DeclareDocumentEnvironment{MacroCode}{m}{
116     \iftoggle{skdoc@impl}{
117         \vspace{.2\baselineskip}
118         \par\noindent
119     }{}
120     \skdoc@verbatim{#1}
121 }{
122     \endskdoc@verbatim
123     \iftoggle{skdoc@impl}{
124         \vspace{.5\baselineskip}

```



```

125     }{}
126 }

```

3.4.1 Reading a preamble

`\PreambleTo` #1: A token to which we will save the preamble
 #2: File to read the preamble from
 Read preamble from a document and store in variable.

```

127 \DeclareDocumentCommand\PreambleTo{mm}{%
128     \group_begin:
129     \msg_info:nnn{skdoc}{read-preamble}{#2}
130     \ior_open:Nn\skdoc@input{#2}
131     \bool_until_do:nn{\ior_if_eof_p:N\
      skdoc@input}{%
132         \tl_if_empty:NTF\skdoc@temptl}{{%
133             \tl_put_right:Nx\skdoc@temptl{\iow_
              newline:}
134         }
135         \tl_clear:N\l_tmpb_tl
136         \ior_get_str:NN\skdoc@input\l_tmpa_tl
137         \tl_put_right:Nx\l_tmpb_tl{\tl_head:N\l_
          tmpa_tl}
138         \IfStrEq{\tl_to_str:N\l_tmpb_tl}{\
          @percentchar}{%
139             \tl_set_eq:NN\l_tmpb_tl\skdoc@temptl
140             \tl_concat:NNN\skdoc@temptl\l_tmpb_
              tl\l_tmpa_tl
141         }{%
142             \ior_close:N\skdoc@input
143         }
144     }
145     \xdef#1{\tl_to_str:N\skdoc@temptl}
146     \group_end:
147 }

```

`\SelfPreambleTo` #1: A token to which we will save the preamble
Shorthand to read preamble from current document.

```
148 \DeclareDocumentCommand\SelfPreambleTo{m}{%  
149     \PreambleTo{#1}{\c_job_name_tl}%  
150 }
```

3.5 Styling

3.5.1 Colors

First, we redefine a couple of colors from ydoc as well as defining a couple for ourselves.

```
151 \providecolorset{RGB}{}{}{  
152     section,11,72,107;  
153     extlink,73,10,61;  
154     intlink,140,35,24;  
155     sharp,250,105,0;  
156     bright,198,229,217;  
157     macrodesc,73,10,61;  
158     keydesc,140,35,24;  
159     macroimpl,73,10,61;  
160     meta,11,72,107;  
161     scriptcolor,140,35,24;  
162     optioncolor,250,105,0;  
163     opt,250,205,0  
164 }
```

3.5.2 Fonts

Then we redefine a couple of the KOMA-script font commands to use our newly defined colors, along with other fixes.

```
165 \RenewDocumentCommand\descfont{}{\sffamily\✓  
    fontseries{sb}}
```

```

166 \RenewDocumentCommand\sectfont{}{\sffamily\
    fontseries{sb}}
167 \addtokomafont{minisec}{\bfseries}
168 \addtokomafont{paragraph}{\color{section}}
169 \addtokomafont{section}{\color{section}}
170 \addtokomafont{subsection}{\color{section}}
171 \addtokomafont{subsubsection}{\color{section}}
172 \addtokomafont{descriptionlabel}{\color{section}
    }}
173 \addtokomafont{sectionentry}{\rmfamily\bfseries}
174 \addtokomafont{sectionentrypagenumber}{\rmfamily
    \bfseries}

```

3.5.3 Configuring hyperref

Finally, we set up hyperref to also use our colors.

```

175 \hypersetup{
176     colorlinks=true,
177     linkcolor=intlink,
178     anchorcolor=intlink,
179     citecolor=black,
180     urlcolor=extlink
181 }

```

3.6 Documentation macros

We can now start defining the documentation macros.

3.6.1 Inline referencing

We introduce a couple of macros for referencing various constructs in running text, *i.e.* `\cs`-like macros.

`\cs` #1: The macro name to be typeset
The `\cs` macro typesets a macro.

```
182 %\DeclareDocumentCommand\cs{m}{\texttt{\textbackslash
    textbackslash #1}}
```

\env #1: The environment name to be typeset
The **\env** macro typesets an environment.

```
183 %\DeclareDocumentCommand\env{m}{\texttt{#1}}
```

\pkg #1: The package name to be typeset
The **\pkg** macro typesets a package.

```
184 \DeclareDocumentCommand\pkg{m}{\textsf{#1}}
```

\opt #1: The option name to be typeset
The **\opt** macro typesets an option

```
185 \DeclareDocumentCommand\opt{m}{\texttt{#1}}
```

\bib #1: The **BIB_TE_X** entry type name to be typeset
The **\bib** macro typesets a **BIB_TE_X** entry type.

```
186 \DeclareDocumentCommand\bib{m}{\texttt{@#1}}
```

\thm #1: The theme name to be typeset
The **\thm** macro typesets a theme of some sort.

```
187 \DeclareDocumentCommand\thm{m}{\textrm{#1}}
```

3.6.2 Descriptive macros

A range of descriptive macros are made available by the **ydoc** package, but we need to redefine and extend them.

We begin with extending.

\Describe@Macro #1: The macro name, including leading backslash
The **\Describe@Macro** macro is changed to typeset its argument in a **\marginnote** instead of an **\fbox**.

```

188 \def\DescribeMacro#1{%
189     \endgroup
190     \edef\name{\expandafter\@gobble\string#1}%
191     \global\@namedef{href@desc@\name}{}%
192     \immediate\write\@mainaux{%
193         \global\noexpand\@namedef{href@desc@\name}{}%
194     }%
195     \hbox\y@bgroup
196     \@ifundefined{href@impl@\name}{}{\hyperlink{
197         impl:\name}}%
198     {%
199         \hbox{
200             \vbox to 0pt{
201                 \vss\hbox{
202                     \raisebox{4ex}{\hypertarget{
203                         desc:\name}}{}
204                 }
205             }%
206             \marginnote{\llap{\PrintMacroName{#1}}}%
207         }%
208     }%
209     \ydoc@macrocatcodes
210     \macroargsstyle
211     \read@Macro@arg
212 }

```

\descframe #1: Contents to be framed

Similarly, \descframe is changed to produce an \mbox instead of an \fbox.

```

212 \def\descframe#1{%
213     \mbox{\hspace*{1.5\descsep}#1\hspace*{2\descsep}}%
214 }

```

`\PrintMacroName` #1: Macro name to be printed
`\PrintMacroName` is hooked to also index the macro name while printing it.

```
215 \let\old@PrintMacroName\PrintMacroName
216 \DeclareDocumentCommand\PrintMacroName{m}{%
217     \index@macro{#1}
218     \old@PrintMacroName{#1}
219 }
```

`\PrintEnvName` #1: Either `\end` or `\begin`
#2: Name of the environment to be printed
Similarly to `\PrintMacroName`, the `\PrintEnvName` is hooked to index the environment when printing the `\begin` part of the environment.

```
220 \let\old@PrintEnvName\PrintEnvName
221 \def\PrintEnvName#1#2{
222     \ifx#1\begin
223         \edef\skdoc@temp{\noexpand\
224             index@environment{#2}}
225         \skdoc@temp
226     \fi
227     \old@PrintEnvName{#1}{#2}
228 }
```

Then we add a few of our own. For instance, we add macros to typeset descriptions of options. We also undefine the `\optpar` macro defined by `ydoc`, since we supersede it with `\Option`.

```
228 \let\optpar\relax
```

`\Options` #1: A comma-separated list of options

```
229 \DeclareDocumentCommand\Options{m}{
230     \clist_set:Nn\l_tmpa_clist{#1}
231     \marginnote{
232         \clist_map_inline:Nn\l_tmpa_clist{
```

```

233         \index@option{####1}
234         \hfill
235         \llap{\textcolor{opt}{\opt{####1}}}{\textcolor{opt}{\opt{####1}}}
236         \mbox{}\\
237     }
238 }
239 \nobreak
240 }

```

\Option #1: And option

```

241 \DeclareDocumentCommand\Option{m}{
242     \Options{#1}
243 }
244 \def\skdoc@WithValues@peek{
245     \ifx\@let@token\AndDefault\else\par\nobreak\✓
246     fi
247 }

```

\WithValues #1: Values of a key-value option

The `\WithValues` macro peeks ahead to see if there's an `\AndDefault` macro further down. It typesets the values of a key-value option

```

247 \DeclareDocumentCommand\WithValues{m}{
248     \noindent\makebox[\linewidth][l]{\texttt{\textcolor{gray}{#1}}}
249     \futurelet\@let@token\skdoc@WithValues@peek
250 }

```

\AndDefault #1: The value of a key-value option

Typesets the default value of a key-value option. To the far right of the line.

```

251 \DeclareDocumentCommand\AndDefault{m}{
252     \llap{\textcolor{gray}{\texttt{(#1)}}}\par\✓
253     nobreak
254 }

```

```
253 }
```

Lastly, we define an environment for showing examples. It's ridiculously simple, really.

example (no arguments)

```
254 \DeclareDocumentEnvironment{example}{}{
255     \minisec{Example:}
256 }
```

3.6.3 Implementation environment

We define environments that encase the implementation of macros, environments, options, \LaTeX entry types and themes. Watch out—there's a lot of duplicate code here.

macro #1: True if this is the starred variant
#2: Name of the macro being implemented
#3: If given, the number of arguments that `\macro@impl@args` will read

```
257 \DeclareDocumentEnvironment{macro}{smo}{%
258     \@bsphack
259     \index@macro*{#2}
260     \@macroname{#2}%
261     \IfBooleanTF{#1}{}{
262         \PrintMacroImpl{#2}
263         \IfNoValueTF{#3}{
264             \macro@impl@argline@noarg{(no~
                arguments)}
265             }\macro@impl@args[#3]}
266     }%
267 }{
268     \let\skdoc@macroname@key\@empty
269     \@esphack
270 }
```

environment #1: True if this is the starred variant
#2: Name of the environment being implemented

#3: If given, the number of arguments that `\macro@impl@args` will read

```
271 \DeclareDocumentEnvironment{environment}{smo}{%
272     \@bsphack
273     \index@environment*{#2}
274     \@environmentname{#2}%
275     \IfBooleanTF{#1}{}{
276         \PrintEnvImplName{#2}
277         \IfNoValueTF{#3}{
278             \macro@impl@argline@noarg{(no~✓
                arguments)}
279         }{\macro@impl@args[#3]}
280     }%
281 }{
282     \let\skdoc@macroname@key\@empty
283     \@esphack
284 }
```

option #1: True if this is the starred variant
#2: Name of the option being implemented
#3: Values this key-value option can take

```
285 \DeclareDocumentEnvironment{option}{smg}{%
286     \@bsphack
287     \index@option*{#2}
288     \@optionname{#2}%
289     \IfBooleanTF{#1}{}{
290         \PrintEnvImplName{#2}
291         \IfNoValueTF{#3}{
292             \macro@impl@argline@noarg{(option)}
293         }{
294             \macro@impl@argline@noarg{
295                 Option~with~values~\texttt{\✓
                    textcolor{gray}{#3}}
296             }
297         }
298     }%
299 }
```

```

300     \let\skdoc@macroname@key\@empty
301     \@esphack
302 }

bibentry #1: True if this is the starred variant
          #2: Name of the BibTeX entry type being implemented

303 \DeclareDocumentEnvironment{bibentry}{sm}{%
304     \@bsphack
305     \index@bibentry*{#2}
306     \@bibentryname{#2}%
307     \IfBooleanTF{#1}{%{
308         \PrintEnvImplName{#2}
309         \macro@impl@argline@noarg{(\hologoRobust
310             {BibTeX}~entry~type)}
311     }%
312 }{
313     \let\skdoc@macroname@key\@empty
314     \@esphack
315 }

theme #1: True if this is the starred variant
        #2: Name of the theme being implemented

315 \DeclareDocumentEnvironment{theme}{sm}{%
316     \@bsphack
317     \index@theme*{#2}
318     \@themenamename{#2}%
319     \IfBooleanTF{#1}{%{
320         \PrintEnvImplName{#2}
321         \macro@impl@argline@noarg{(theme)}
322     }%
323 }{
324     \let\skdoc@macroname@key\@empty
325     \@esphack
326 }

```

We also provide starred variants of the environments, which will add the implementation to the index but not print anything.

macro* #1: Name of the macro being implemented
 #2: If given, the number of arguments that `\macro@impl@args` will read

```
327 \DeclareDocumentEnvironment{macro*}{mo}%
328   {\begin{macro}*{#1}[#2]}\end{macro}}
```

environment* #1: Name of the environment being implemented
 #2: If given, the number of arguments that `\macro@impl@args` will read

```
329 \DeclareDocumentEnvironment{environment*}{mo}%
330   {\begin{environment}*{#1}[#2]}\end{environment}}
```

option* #1: Name of the option being implemented
 #2: Values this key-value option can take

```
331 \DeclareDocumentEnvironment{option*}{mg}%
332   {\begin{option}*{#1}{#2}}\end{option}}
```

bibentry* #1: Name of the BibTeX entry type being implemented

```
333 \DeclareDocumentEnvironment{bibentry*}{m}%
334   {\begin{bibentry}*{#1}}\end{bibentry}}
```

theme* #1: Name of the theme being implemented

```
335 \DeclareDocumentEnvironment{theme*}{m}%
336   {\begin{theme}*{#1}}\end{theme}}
```

And finally, we redefine some of the underlying ydoc macros to behave the way we want them to. For instance, we redefine the commands that print environment and macro implementation names so that they typeset the name in a `\marginnote` rather than in an `\fbox`.

\PrintEnvImplName #1: The environment name to be printed

```
337 \def\PrintEnvImplName#1{%
338   \par\mbox{}
339   \marginnote{\llap{\implstyle{#1}}}%
340   \par
341 }
```

`\PrintMacroImpl` #1: The macro name to be printed

```
342 \def\PrintMacroImpl#1{%
343   \par
344   \hbox{%
345     \edef\name{\expandafter\@gobble\string
346               #1}%
347     \global\@namedef{href@impl@\name}{}%
348     \immediate\write\@mainaux{%
349       \global\noexpand\@namedef{href@impl@
350       \name}{}%
351     }%
352     \raisebox{4ex}[4ex]{\hypertarget{impl:\
353     name}{}}%
354     \@ifundefined{href@desc@\name}{}{\
355     hyperlink{desc:\name}}%
356     \marginnote{\llap{\PrintMacroImplName
357     {#1}}}%
358   }%
359   \par
360 }
```

We also redefine the utility macros belonging to `\macro@impl@arg`.
The argument number Description of the argument

```
356 \def\macro@impl@argline#1#2{%
357   \par\noindent{\texttt{\##1}:~#2\strut}%
358 }
```

The number of arguments to read

```
359 \def\macro@impl@args[#1]{%
360   \vspace*{-\baselineskip}
361   \begingroup
362   \let\macro@impl@argcnt\@tempcnta
363   \let\macro@impl@curarg\@tempcntb
364   \macro@impl@argcnt=#1\relax
365   \macro@impl@curarg=0\relax
```

```

366     \ifnum\macro@impl@curarg<\macro@impl@argcnt\relax
           relax
367     \expandafter\macro@impl@arg
368 \else
           \expandafter\macro@impl@endargs
369 \fi
370 }
371 }

372 \def\macro@impl@endargs{
373     \endgroup\par
374 }

```

The line to print instead of an argument line This last macro is a replacement used when there are no arguments or if the implementation is an option or something like that. It behaves pretty much like `\macro@impl@args`, but with only one argument to read.

```

375 \def\macro@impl@argline@noarg#1{%
376     \vspace*{-\baselineskip}
377     \par\noindent{#1\strut}\par\medskip%
378 }

```

3.7 The index

The index is based on glossaries, and as such the whole process of adding entries to the index is reduced to adding glossary entries. This is done through two wrapper macros around the `\newglossaryentry` macro.

`\@index@` #1: The key of the index entry

The text of the index entry What `\@index@` does is to decide whether we are hiding the implementation part of the documentation (discussed later), and whether we are in the actual implementation or not. If we are in the implementation and aren't printing it, we shouldn't add an index entry.

```

379 \DeclareDocumentCommand\@index@{mm}{
380     \iftoggle{skdoc@impl}{

```

```

381         \@index@@{#1}{#2}
382     }{
383         \iftoggle{skdoc@in@impl}{}{
384             \@index@@{#1}{#2}
385         }
386     }
387 }

```

\@index@@ #1: The key of the index entry
#2: The text of the index entry
This macro does the actual adding to the glossary.

```

388 \DeclareDocumentCommand\@index@@{mm}{
389     \newglossaryentry{index-#1}{
390         type=index ,
391         name={#2} ,
392         description={\nopostdesc} ,
393         sort={#1}
394     }
395 }

```

3.7.1 Adding index entries

These macros add an index entry with different contents depending on the thing (macro, environment, etc.) that is being indexed. They all have starred variants which are used by the implementation environments, and non-starred variants used by the description macros (the star affects the style of the page number). Each environment first test wether the given entry key exists, and defines a new entry if it doesn't. Then, a usage of the entry is recorded.

\index@macro #1: True if this is the starred variant
#2: The name of the macro being indexed, including backslash

```

396 \DeclareDocumentCommand\index@macro{sm}{
397     \def\skdoc@temp{\expandafter\@gobble\string
398         #2}

```

```

398     \ifglentryexists{index-\skdoc@temp}}}{
399         \@index@{\expandafter\@gobble\string#2}
400         {\cs{\expandafter\@gobble\string
401             #2}}
402     }
403     \IfBooleanTF{#1}{
404         \glsadd[types=index]{index-\skdoc@temp}
405     }{
406         \glsadd[types=index,format=hyperit]{\
407             index-\skdoc@temp}
408     }
409 }

```

\index@environment #1: True if this is the starred variant
#2: The name of the environment being indexed

```

408 \DeclareDocumentCommand\index@environment{sm}{
409     \def\skdoc@temp{\string#2}
410     \ifglentryexists{index-\skdoc@temp}}}{
411         \@index@{\string#2}
412         {\env{\string#2}~(environment)}
413     }
414     \IfBooleanTF{#1}{
415         \glsadd[types=index]{index-\skdoc@temp}
416     }{
417         \glsadd[types=index,format=hyperit]{\
418             index-\skdoc@temp}
419     }
420 }

```

\index@option #1: True if this is the starred variant
#2: The name of the option being indexed

```

420 \DeclareDocumentCommand\index@option{sm}{
421     \def\skdoc@temp{\string#2}
422     \ifglentryexists{index-\skdoc@temp}}}{

```

```

423         \@index@{\string#2}
424         {\opt{\string#2}~(option)}
425     }
426     \IfBooleanTF{#1}{
427         \glsadd[types=index]{index-\skdoc@temp}
428     }{
429         \glsadd[types=index,format=hyperit]{\sk
430         index-\skdoc@temp}
431     }

```

\index@bibentry #1: True if this is the starred variant
#2: The name of the BibTeX entry type being indexed, including initial @ sign

```

432 \DeclareDocumentCommand\index@bibentry{sm}{
433     \def\skdoc@temp{\expandafter\@gobble\string
434     #2}
435     \ifglentryexists{index-\skdoc@temp}{}{
436         \@index@{\expandafter\@gobble\string#2}
437         {\bib{\expandafter\@gobble\string#2}~(\holoRobust{
438         BibTeX}~entry~type)}
439     }
440     \IfBooleanTF{#1}{
441         \glsadd[types=index]{index-\skdoc@temp}
442     }{
443         \glsadd[types=index,format=hyperit]{\sk
444         index-\skdoc@temp}
445     }

```

\index@theme #1: True if this is the starred variant
#2: The name of the theme being indexed

```

444 \DeclareDocumentCommand\index@theme{sm}{

```



```

445 \def\skdoc@temp{\string#2}
446 \ifglentryexists{index-\skdoc@temp}{}{
447   \@index@{\string#2}
448   {\thm{\string#2}~(theme)}
449 }
450 \IfBooleanTF{#1}{
451   \glsadd[types=index]{index-\skdoc@temp}
452 }{
453   \glsadd[types=index,format=hyperit]{\skdoc@temp}
454 }
455 }

```

3.7.2 Displaying the index

Displaying the index is very simple. We begin by defining our own glossaries style.

```

456 \newglossarystyle{docindex}{
457   \glossarystyle{indexgroup}
458   \renewcommand*{\glspostdescription}{\unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}
459   \renewenvironment{theglossary}{
460     \begin{multicols}{2}
461     \setlength{\parindent}{0pt}
462     \setlength{\parskip}{0pt plus 0.3pt}
463     \let\item\@idxitem
464   }{
465     \end{multicols}
466   }
467   \renewcommand*{\glossaryentryfield}[5]{
468     \item\glstentryitem{##1}\glstarget{##1}{##2}
469     \ifx\relax##4\relax\else\space(##4)\fi
470     ##3\glspostdescription\space ##5}

```

```

471     \renewcommand*{\glsgroupheading}[1]{
472         \IfStrEq{##1}{default}{
473             \item{\descfont\glssymbolsgroupname}
474         }{
475             \item{\descfont\glsgetgrouptitle
476                 {##1}}
477         }
478     \renewcommand*{\glsgroupskip}{
479         \par\vspace{15\p@}\relax
480     }
481 }

```

We follow that up by defining the actual glossary, and making sure to run `\makeglossaries` when the preamble is complete.

```

482 \newglossary{index}{ids}{ido}{Index}
483 \AtBeginDocument{\makeglossaries}

```

`\PrintIndex` (no arguments)

Finally, we define a command `\PrintIndex` that prints the index. Note the preamble that describes how the index page numbers work.

```

484 \providecommand*\PrintIndex{%
485     \begingroup
486     \renewcommand*{\glossarypreamble}{
487         Numbers~written~in~italic~refer~to~the~
488         page~where~the~
489         corresponding~entry~is~described;~
490         numbers~in~roman~
491         refer~to~the~code~line~of~the~definition~
492         .\par
493     }
494     \printglossary[type=index,style=docindex]
495     \endgroup
496 }

```

3.8 The changelog

The changelog is implemented as a glossary using the glossaries package. We begin by defining a name for general changes, and commands that save the name of the current macro, environment or similar for use by the `\changes` macro.

`\generalname` (no arguments)

```
494 \DeclareDocumentCommand\generalname{}{General}
      Name of the macro being described
495 \DeclareDocumentCommand\@macroname{m}{
496     \def\skdoc@macroname@stylized{\cs{\
      expandafter\@gobble\string#1}}
497     \def\skdoc@macroname@key{macro-\expandafter\
      @gobble\string#1}
498 }
      Name of the environment being described
499 \DeclareDocumentCommand\@environmentname{m}{
500     \def\skdoc@macroname@stylized{\env{\string
      #1}}
501     \def\skdoc@macroname@key{env-#1}
502 }
      Name of the option being described
503 \DeclareDocumentCommand\@optionname{m}{
504     \def\skdoc@macroname@stylized{\opt{\string
      #1}}
505     \def\skdoc@macroname@key{opt-#1}
506 }
      Name of the BBTEX entry being described
507 \DeclareDocumentCommand\@bibentryname{m}{
508     \def\skdoc@macroname@stylized{\bib{\
      expandafter\@gobble\string#1}}
```

```

509     \def\skdoc@macroname@key{bibentry-\✓
        expandafter\@gobble\string#1}
510 }

    Name of the theme being described

511 \DeclareDocumentCommand\@themenamem{m}{
512     \def\skdoc@macroname@stylized{\thm\string✓
        #1}}
513     \def\skdoc@macroname@key{thm-#1}
514 }

    Along with these we also define the variables they affect as empty.

515 \def\skdoc@macroname@stylized{}
516 \let\skdoc@macroname@key\@empty

```

3.8.1 Adding changes

Since the changelog is based on glossaries, adding changes amounts to simply adding a glossary entry.

```

\changes #1: The version in which the changes were made
        #2: A short description of the changes

517 \DeclareDocumentCommand\changes{mm}{%
518     \@bsphack
519     \ifglentryexists{#1}{}{
520         \newglossaryentry{#1}{
521             type=changes ,
522             name={v#1} ,
523             description={\nopostdesc} ,
524             nonumberlist=true
525         }
526     }
527     \ifx\skdoc@macroname@key\@empty
528         \newglossaryentry{#1-general}{
529             type=changes ,

```

```

550         description={\generalname{}}:~#2},
551         parent={#1},
552         sort={0},
553         nonumberlist=true
554     }
555     \glsadd[types=changes]{#1-general}
556 \else
557     \newglossaryentry{#1-\↙
558         skdoc@macroname@key}{
559         type=changes ,
560         description={\↙
561             skdoc@macroname@stylized{}}:~#2},
562         parent={#1},
563         sort={\skdoc@macroname@key},
564         nonumberlist=true
565     }
566     \glsadd[types=changes]{#1-\↙
567         skdoc@macroname@key}
568 \fi
569 \@esphack
570 }

```

3.8.2 Displaying the changelog

Displaying the changelog is equally simple. We begin by defining our own glossaries style.

```

548 \newglossarystyle{changelog}{
549     \glossarystyle{altlist}
550     \renewenvironment{theglossary}{
551         \begin{multicols}{2}\begin{description}
552     }{
553         \end{description}\end{multicols}
554     }
555     \renewcommand*{\glossaryentryfield}[5]{
556         \par\vspace{5\p@}\relax

```

```

557         \item[\glsubentryitem{##1}\glstarget{
           {##1}{##2}}]
558             \mbox{}\par\nobreak\
               @afterheading
559     }
560     \renewcommand{\glossarysubentryfield}[6]{%
561         \par\hspace*{\itemindent}
562         \glssubentryitem{##2}%
563         \glstarget{##2}{\strut}##4\
           glspostdescription\space ##6
564     }
565 }

```

We follow that up by defining the actual glossary, and making sure to run `\makeglossaries` when the preamble is complete.

```

566 \newglossary{changes}{gls}{glo}{Changes}
567 \AtBeginDocument{\makeglossaries}

```

`\PrintChanges` (no arguments)

Finally, we define a command `\PrintChanges` that prints the list of changes.

```

568 \DeclareDocumentCommand\PrintChanges{}{%
569     \begingroup
570     \printglossary[type=changes,style=changelog]
571     \endgroup
572 }

```

3.9 Hiding the implementation

We define commands to hide the implementation from the documentation. Here, the “implementation” is understood to be everything between the `\Implementation` and `\Finale` macros. What we do is disable and/or reset page and section counters for the duration of the implementation, and set a shipout hook that simply discards the pages while we are in

the implementation. A consequence of this is that we must force a page break between what's before the implementation and what's after, which might look horrible.

Anyway, we define two toggles that keep track of things. One keeps track of whether to (not) hide the implementation, and one keeps track of whether we are in the implementation or not. These are provided with sensible defaults (*i.e.* true and false, respectively). We also define a counter in which we save the page number we had when the implementation started.

```
573 \newtoggle{skdoc@impl}
574 \newtoggle{skdoc@in@impl}
575 \toggletrue{skdoc@impl}
576 \togglefalse{skdoc@in@impl}
577 \newcounter{skdoc@impl@page}
```

Then we define the shipout hook. Fairly straight-forward.

```
578 \AtBeginShipout{
579     \iftoggle{skdoc@impl}{}{
580         \iftoggle{skdoc@in@impl}{\✓
581             AtBeginShipoutDiscard}{}
582     }
```

\OnlyDescription (no arguments)

The `\OnlyDescription` macro simply toggles the appropriate toggle.

```
583 \DeclareDocumentCommand\OnlyDescription{}{\✓
    togglefalse{skdoc@impl}}
```

\Implementation (no arguments)

The `\Implementation` macro defines all the sectioning commands to be empty (saving the old ones), clears the page, saves the page number and toggles the appropriate toggle.

```

584 \DeclareDocumentCommand\Implementation{}{
585     \iftoggle{skdoc@impl}{}{
586         \clearpage
587         \toggletrue{skdoc@in@impl}
588         \let\skdoc@old@part\part
589         \DeclareDocumentCommand\part{som}{}
590         \let\skdoc@old@section\section
591         \DeclareDocumentCommand\section{som}{}
592         \let\skdoc@old@subsection\subsection
593         \DeclareDocumentCommand\subsection{som↵
594             }{}
595         \let\skdoc@old@subsubsection↵
596             subsubsection
597         \DeclareDocumentCommand\subsubsection{↵
598             som}{}
599         \let\skdoc@old@paragraph\paragraph
600         \DeclareDocumentCommand\paragraph{som}{}
601         \let\skdoc@old@subparagraph\subparagraph
602         \DeclareDocumentCommand\subparagraph{som↵
603             }{}
604         \setcounter{skdoc@impl@page}{\value{page↵
605             }}
606     }
607 }

```

\Finale (no arguments)

The `\Finale` macro basically just undoes what the `\Implementation` macro did.

```

603 \DeclareDocumentCommand\Finale{}{
604     \iftoggle{skdoc@impl}{}{
605         \clearpage
606         \togglefalse{skdoc@in@impl}
607         \let\part\skdoc@old@part
608         \let\section\skdoc@old@section
609         \let\subsection\skdoc@old@subsection

```



```

610         \let\subsubsection\✓
           skdoc@old@subsubsection
611     \let\paragraph\skdoc@old@paragraph
612     \let\subparagraph\skdoc@old@subparagraph
613     \setcounter{page}{\value{skdoc@impl@page}✓
        }}
614   }
615 }

```

3.10 Document metadata

3.10.1 Setting metadata

We override a bunch of the general titlepage macros and add a few of our own. First, we initialize the underlying variables.

```

616 \let\@ctan\@empty%
617 \let\@repository\@empty%
618 \let\@plainemail\@empty%
619 \let\@email\@empty%
620 \let\@version\@empty%

```

Then, we define the actual macros.

`\package` #1: A list of key-value options

#2: The package name

The `\package` macro sets the package name of the documentation. The key-value options are `vcs` and `ctan`.

```

621 \DeclareDocumentCommand\package{om}{%
622   \keys_define:nn{skdoc@package}{%
623     vcs .value_required:,%
624     vcs .code:n = \repository{##1},%
625     ctan .code:n = \ctan{##1},%
626     ctan .default:n = #2%
627   }%
628   \IfNoValueTF{#1}{}{\keys_set:nn{✓
     skdoc@package}{#1}}%

```

```

629     \def\@package{#2}%
630     \title{The~\textsf{\textbf{\@package}}~\textbf{\@package}}%
631 }

```

\ctan #1: The name of a package or bundle on CTAN

```

632 \DeclareDocumentCommand\ctan{m}{%
633     \def\@ctan{\url{http://www.ctan.org/pkg/#1}}%
634 }

```

\repository #1: The URI of an online repository

```

635 \DeclareDocumentCommand\repository{m}{%
636     \def\@repository{\url{#1}}%
637 }

```

\email #1: The email address of the author

```

638 \DeclareDocumentCommand\email{m}{%
639     \def\@plainemail{#1}%
640     \def\@email{\href{mailto:\@plainemail}{\@plainemail}}%
641 }

```

\version #1: The version of the package, with no leading “v”

```

642 \DeclareDocumentCommand\version{m}{%
643     \def\@version{#1}%
644 }

```

Finally, we set the default package name to \jobname.

```

645 \package{\jobname}

```

3.10.2 Using metadata

We define two macros that read useful metadata; `\theversion` and `\thepackage`. These are used internally by `\maketitle`.

`\theversion` (no arguments)

```
646 \DeclareDocumentCommand\theversion{}{v\@version}
```

`\thepackage` (no arguments)

```
647 \DeclareDocumentCommand\thepackage{}{\pkg{\@package}}
```

Additionally we define `\skdocpdfsettings`, which is also used by `\maketitle`, to include PDF metadata if the documentation is being compiled using pdf~~L~~TeX.

```
648 \ifpdf
649   \def\skdocpdfsettings{%
650     \hypersetup{%
651       pdfauthor    = {\@author\space<\@
652         @plainemail>},
653       pdftitle     = {\@title},
654       pdfsubject   = {Documentation~of~
655         LaTeX~package~\@package},
656       pdfkeywords  = {\@package,~LaTeX,~TeX}
657     }%
658   }%
659 \else
660   \let\skdocpdfsettings\empty%
661 \fi
```

3.11 General document commands

Most of the general document commands are defined by the `scrartcl` document class we base ourselves on, but a few of them have to be redefined.

3.11.1 The title page

The title page consists of the `\maketitle` and the abstract. We redefine both, inspired slightly by the `PracTEX` journal and the `skrapport` document class.

`\@maketitle` (no arguments)

```
660 \def\@maketitle{%
661     \newpage
662     \null
663     \begin{flushleft}%
664     {%
665         \Huge\sectfont\@title%
666         \ifx\@ctan\@empty\else%
667             \footnote{Available~on~\@ctan.}%
668         \fi
669         \ifx\@repository\@empty\else%
670             \footnote{Development~version~✓
671                 available~on~\@repository.}%
672         \fi%
673     }%
674     \vskip 1em
675     {%
676         \Large\@author
677         \ifx\@email\@empty\else%
678             \space
679             \newlength\skdoc@minipage@ew%
680             \settowidth{\skdoc@minipage@ew}{%
```

```

681             \normalsize{$\lceil$\@email$\rceil$}
682             \begin{minipage}[b]{\skdoc@minipage@ew}
683             \normalsize{$\lceil$\@email$\rceil$}
684             \end{minipage}\par%
685     \fi%
686 }%
687 \ifx\@version\@empty\else
688     \vskip .5em
689     {%
690         \large Version~\@version\par%
691     }%
692 \fi
693 \end{flushleft}%
694 \par\bigskip%
695 }

```

\maketitle (no arguments)

```

696 \def\maketitle{%
697     \skdocpdfsettings
698     \renewcommand\thefootnote{\@fnsymbol\c@footnote}
699     \@maketitle
700     \skdocpdfsettings
701 }

```

abstract (no arguments)

```

702 \DeclareDocumentEnvironment{abstract}{}{
703     \newlength\skdoc@abstract@tw%
704     \newlength\skdoc@abstract@aw%
705     \settowidth{\skdoc@abstract@tw}{\descfont\abstractname}%

```

```

706 \setlength{\skdoc@abstract@aw}{\the\✓
      textwidth-\the\skdoc@abstract@tw-2em}%
707 \begin{minipage}{\textwidth}
708   \begin{minipage}[t]{\skdoc@abstract@tw}%
709     \begin{flushright}%
710       \leavevmode\descfont\✓
          abstractname%
711     \end{flushright}%
712   \end{minipage}%
713   \hspace{1em}%
714   \begin{minipage}[t]{\skdoc@abstract@aw}%
715 }{
716   \end{minipage}
717 \end{minipage}
718 }

```

3.11.2 Table of contents

The table of contents are redefined to imitate the excellent table of contents of the microtype manual.

```

719 \let\l@section@\l@section
720 \def\l@section{\vskip -.75ex\l@section@}

721 \def\l@subsection{\vskip.35ex\penalty\✓
      @secpenalty\@dottedtocline{2}{1.5em}{2.7em}}

722 \def\l@subsubsection#1#2{
723   \leftskip 4.2em
724   \parindent 0pt
725   {\let\numberline\@gobble{\small #1~[#2]}}
726 }

727 \def\l@table{\@dottedtocline{1}{0pt}{1.5em}}

728 \def\l@figure{\@dottedtocline{1}{0pt}{1.5em}}

```

```

729 \def\@pnumwidth{1.7em}
730 \AtEndDocument{\addtocontents{toc}{\par}}

```

`\tableofcontents` (no arguments)

```

731 \let\old@tableofcontents\tableofcontents
732 \DeclareDocumentCommand\tableofcontents{}{
733     \microtypesetup{protrusion=false}
734     \old@tableofcontents
735     \microtypesetup{protrusion=true}
736 }

```

3.12 Cosmetic changes

We perform a couple of cosmetic changes to existing features as well. First, we set a new header/footer style using the KOMA-script `\deftripstyle` macro.

```

737 \deftripstyle{skdoc}%
738     {}{}{}%
739     {\small The~\textsf{\textbf{\@package}}~✓
       package,~v\@version}{}{\small\pagemark}
740 \AfterBeginDocument{\pagestyle{skdoc}}

```

We also redefine the section level format to set the section numbers in the margin, much like the microtype manual.

```

741 \RenewDocumentCommand{\othersectionlevelsformat}✓
       {}{m}{%
742     \makebox[0pt][r]{%
743     \fontfamily{fos}\mdseries\selectfont
744     \csname the#1\endcsname\enskip}%
745 }

```

Finally, we actually use microtype in the document class, and make sure to disable it in the verbatim environments. Set up microtype properly

```

746 \g@addto@macro\@verbatim{\microtypesetup{✓
      activate=false}}
747 \microtypesetup{expansion,kerning,spacing,✓
      tracking}
748 \DisableLigatures{family = tt*}
      Oh, and we want \marginpars on the left, not on the right.
749 \AtBeginDocument{\reversemarginpar}
      That's it, we're done!
750 \endinput

```

4 Changes

vo.9 ality implemented.

General: Pre-release version with
most of the general function-

5 Index

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

Symbols	A
\@ebibentryname 27	abstract (environment) . . . 37
\@environmentname 27	\AndDefault 15
\@index@ 21	B
\@index@@ 22	\bib 12
\@macroname 27	bibentry (environment) . . . 18
\@maketitle 36	bibentry* (environment) . . 19
\@optionname 27	C
\@subsubsection 38	\changes 28
\@themenam 28	

<code>\cs</code>	11	M	
<code>\ctan</code>	34		<code>macro (environment)</code> 16
			<code>macro* (environment)</code> 18
D			<code>\macro@impl@argline</code> 20
<code>\DeclareFile</code>	5		<code>\macro@impl@argline@noarg</code> 21
<code>\descframe</code>	13		<code>\macro@impl@args</code> 20
<code>\Describe@Macro</code>	12		<code>\macro@impl@endargs</code> 21
			<code>MacroCode (environment)</code> . . . 8
E			<code>\maketitle</code> 37
<code>\email</code>	34	O	
<code>\env</code>	12		<code>\OnlyDescription</code> 31
<code>environment (environment)</code> .	16		<code>\opt</code> 12
<code>environment* (environment)</code> .	19		<code>\Option</code> 15
<code>example (environment)</code>	16		<code>option (environment)</code> 17
F			<code>option* (environment)</code> 19
<code>\Finale</code>	32		<code>\Options</code> 14
		P	
G			<code>\package</code> 33
<code>\generalname</code>	27		<code>\pkg</code> 12
I			<code>\PreambleTo</code> 9
<code>\Implementation</code>	31		<code>\PrintChanges</code> 30
<code>\index@bibentry</code>	24		<code>\PrintEnvImplName</code> 19
<code>\index@environment</code>	23		<code>\PrintEnvName</code> 14
<code>\index@macro</code>	22		<code>\PrintIndex</code> 26
<code>\index@option</code>	23		<code>\PrintMacroImpl</code> 20
<code>\index@theme</code>	24		<code>\PrintMacroName</code> 14
		R	
L			<code>\repository</code> 34
<code>\l@figure</code>	38	S	
<code>\l@section</code>	38		<code>\SelfPreambleTo</code> 10
<code>\l@subsection</code>	38		<code>skdoc@verbatim (environment)</code> 7
<code>\l@table</code>	38		

<code>\skdoc@WithValues@peek</code> ..	15	<code>\thm</code>	12
T		V	
<code>\tableofcontents</code>	39	<code>\version</code>	34
<code>theme (environment)</code>	18		
<code>theme* (environment)</code>	19	W	
<code>\thepackage</code>	35	<code>\WithValues</code>	15
<code>\theversion</code>	35		