THE CNLTX BUNDLE

Documentation for LaTeX 2_ε Packages or Classes

v0.5   2013/09/18

LaTeX examples the CN way

Clemens NIEDERBERGER

https://github.com/cgnieder/cnltx

contact@mychemistry.eu

A bundle of packages and classes for consistent format of control sequences, package options, source code with examples, writing a package manual (including an index containing the explained control sequences, options, . . .).

Table of Contents

Part I.
About The Bundle

1. Background

The CNLTX bundle contains different packages and classes. I developed it as a successor of my class cnpkgdoc that I used for writing the documentation of my packages. The intention behind this is a cleaner interface and less unnecessary ballast, hence the separation into packages and classes. The bundle provides source code environments that also print the output and defines quite a number of macros for formatting of control sequence names, package names, package options and so on.

Part of the motivation is also that users have asked me how I created the manuals for my packages. Now I can refer to this bundle.

Another reason for the splitting into separate packages is – besides the advantage of easier maintenance – is that I may want to add programming tools that I use often into CNLTX-BASE which may allow me (and others) to use them for other packages, too, without having to define them each time. So it is quite possible that CNLTX-BASE will get extended in the future.

The best documentation for the bundle as always is the source code of the sty and cls files

but I'm trying to provide a documentation as comprehensive as possible. This is one of the reasons why this documentation is noticeably longer than the one for cnpkgdoc.

The bundle reflects the fact that I haven't started using literate programming, yet. I don't use `docstrip` and don't write `dtx` files but always write the `sty` or `cls` files directly. The manual is always created parallel but separately. While I'm entirely aware of the advantages of literate programming I never could bring myself to start to use it myself. As a consequence I have no idea if this bundle can be used for it or not.

Source code formatting is done with the help of the powerful listings package [HM13] by Carsten Heinz and later Brooks Moses, now maintained by Jobst Hoffmann. The only real drawback I have found with it is recognizing starred und un-starred versions of an environment as different keywords. This does not seem to be possible which is why indexing of such environments will lead to wrong page numbers.

The fancy frames of the source code examples are realized with the mdframed package[Dan13] by Marco Daniel, loaded with the option `framemethod` = tikz.

2. Bundled Packages, Classes and Files

The CNLTX bundle currently bundles the following packages and classes:

- CNLTX-BASE – a package that defines base macros for error-messaging, expansion control and tokenlist manipulation. It also provides color definitions and defines a few color schemes for the CNLTX-DOC class. All other packages and classes of the CNLTX bundle load this package. This package can be used stand-alone.

- CNLTX-DOC – a class for writing package manuals. Loads CNLTX-EXAMPLE and CNLTX-TOOLS.

- CNLTX-EXAMPLE – a package that defines macros and environments for describing control sequences and options and for including source code. Loads CNLTX-LISTINGS. This package can be used stand-alone.

Introduced in version 0.4
- CNLTX-LISTINGS – a package that defines the listings language 'BibTeX'. Also defines a list of highlighted control sequence names and environment names, loaded by CNLTX-EXAMPLE. The additional control sequence and environment names used to be defined in CNLTX-CSNAMES. That package got removed and its contents are now provided by CNLTX-LISTINGS. This package can be used stand-alone.

Introduced in version 0.2
- CNLTX-TOOLS – a package that defines tools used by CNLTX-DOC that are unrelated to LaTeX documentation *per se*.

- cnltx.ist – an index style file that is used when the option `add-index` is activated and the option `index-style` is not used.

Introduced in version 0.4
- cnltx.bib – a bibliography file that contains a small but growing number of package entries.

Introduced in version 0.4
- cnltx.bbx, cnltx.cbx and cnltx.dbx – files related to the biblatex style cnltx.

3. License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

The CNLTX-BASE package loads the following packages: pgfopts[1] [Wri11], etoolbox[2] [Leh11], trimspaces[3] [Rob09] and xcolor[4] [Ker07].

The CNLTX-DOC class loads the package with the same name and additionally the following packages: CNLTX-BASE, CNLTX-EXAMPLE, translations [Nie13c], ulem[5] [Ars11], multicol[6] [Mit11], ragged2e[7] [Sch09], marginnote[8] [Koh12] and hyperref[9] [OR12]. It is a wrapper class for the KOMA-Script class scrartcl[10] [KN12].

The CNLTX-EXAMPLE package loads the following packages: CNLTX-BASE, CNLTX-LISTINGS, CNLTX-TOOLS, translations,[11] mdframed[12] [Dan13] and idxcmds[13] [Nie13b].

The CNLTX-LISTINGS package loads CNLTX-BASE and listings[14] [HM13].

The CNLTX-TOOLS package loads CNLTX-BASE and accsupp[15] [Obe01].

Like all of my packages CNLTX implicitly relies on an up to date TeX distribution.

Part II.
Details of Available Commands, Environments and Options

4. Options and Setup

The CNLTX bundle has a number of options. The CNLTX-DOC class only knows a few options (described in section 9.1 on page 19) as *class* options. All other options regardless if they're defined by a package or a class can and should be set with the setup command:

---

1. on CTAN as `pgfopts`: `http://mirrors.ctan.org/macros/latex/contrib/pgfopts/`
2. on CTAN as `etoolbox`: `http://mirrors.ctan.org/macros/latex/contrib/etoolbox/`
3. on CTAN as `trimspaces`: `http://mirrors.ctan.org/macros/latex/contrib/trimspaces/`
4. on CTAN as `xcolor`: `http://mirrors.ctan.org/macros/latex/contrib/xcolor/`
5. on CTAN as `ulem`: `http://mirrors.ctan.org/macros/latex/contrib/ulem/`
6. on CTAN as `multicol`: `http://mirrors.ctan.org/macros/latex/required/tools/multicol/`
7. on CTAN as `ragged2e`: `http://mirrors.ctan.org/macros/latex/contrib/ms/ragged2e/`
8. on CTAN as `marginnote`: `http://mirrors.ctan.org/macros/latex/contrib/marginnote/`
9. on CTAN as `hyperref`: `http://mirrors.ctan.org/macros/latex/contrib/hyperref/`
10. on CTAN as `koma-script`: `http://mirrors.ctan.org/macros/latex/contrib/koma-script/`
11. on CTAN as `translations`: `http://mirrors.ctan.org/macros/latex/contrib/translations/`
12. on CTAN as `mdframed`: `http://mirrors.ctan.org/macros/latex/contrib/mdframed/`
13. on CTAN as `idxcmds`: `http://mirrors.ctan.org/macros/latex/contrib/idxcmds/`
14. on CTAN as `listings`: `http://mirrors.ctan.org/macros/latex/contrib/listings/`
15. on CTAN as `accsupp`: `http://mirrors.ctan.org/macros/latex/contrib/oberdiek/accsupp/`

`\setcnltx{⟨options⟩}`
  setup command for CNLTX.

  The source code environments defined by the CNLTX-EXAMPLE package also have optional arguments that can be used to set the options for the environment locally.

5. Available Commands

5.1. Description of Macros, Environments and Options

provided by CNLTX-EXAMPLE

The commands described in this section all are provided by the CNLTX package. They all are related to the typesetting of provided macros, options and the like.

`\code{⟨arg⟩}`
  Formatting of source code. This is *no* verbatim command. Used internally in the following commands.

`\verbcode⟨delim⟩⟨code⟩⟨delim⟩`

Introduced in version 0.2

A verbatim command that uses the same formatting as the source code example environments. This is a wrapper for `\lstinline` which loads the corresponding style.

`\cs*{⟨name⟩}`
  Format the control sequence ⟨name⟩, `\cs{name}`: `\name`. Adds a corresponding index entry. The starred form does not add an index entry.

`\csidx{⟨name⟩}`
  Adds an index entry but does not typeset the control sequence ⟨name⟩.

`\env*{⟨name⟩}`
  Format the environment ⟨name⟩, `\env{name}`: name. Adds a corresponding index entry with a hint that the entry refers to an environment. The starred form does not add an index entry.

`\envidx{⟨name⟩}`
  Adds an index entry but does not typeset the environment ⟨name⟩.

`\meta{⟨meta⟩}`
  Description of an argument, `\meta{meta}`: ⟨meta⟩.

`\marg{⟨arg⟩}`
  A mandatory argument. ⟨arg⟩ is formatted with `\meta` if it is not blank, `\marg{arg}`: {⟨arg⟩}.

`\Marg{⟨arg⟩}`

Introduced in version 0.2

A mandatory argument. ⟨arg⟩ is formatted with `\code` if it is not blank, `\Marg{arg}`: {arg}.

`\oarg{⟨arg⟩}`
  An optional argument. ⟨arg⟩ is formatted with `\meta` if it is not blank, `\oarg{arg}`: [⟨arg⟩].

5. Available Commands

`\Oarg{⟨arg⟩}`

An optional argument. ⟨*arg*⟩ is formatted with `\code` if it is not blank, `\Oarg{arg}`: `[arg]`.

`\darg{⟨arg⟩}`

An argument with parentheses as delimiters. ⟨*arg*⟩ is formatted with `\meta` if it is not blank, `\darg{arg}`: `(⟨arg⟩)`.

`\Darg{⟨arg⟩}`

An argument with parentheses as delimiters. ⟨*arg*⟩ is formatted with `\code` if it is not blank, `\Darg{arg}`: `(arg)`.

`\sarg`

An optional star argument, `\sarg`: `*`.

`\option*{⟨name⟩}`

An option ⟨*name*⟩, `\option{name}`: `name`. Adds a corresponding index entry. The starred form does not add an index entry.

`\optionidx{⟨name⟩}`

Adds an index entry but does not typeset the option ⟨*name*⟩.

`\module*{⟨name⟩}`

A module ⟨*name*⟩, `\module{name}`: `name`. Adds a corresponding index entry. The starred form does not add an index entry. In some of my package I like to organize options by grouping them in different classes that I call "modules". This command refers to those modules.

`\moduleidx*{⟨name⟩}`

Adds an index entry but does not typeset the option ⟨*name*⟩.

`\key*-{⟨name⟩}{⟨value⟩}`

A key ⟨*name*⟩ with value ⟨*value*⟩, the optional star prevents an index entry, the optional - strips the braces around ⟨*value*⟩; `\key{key}{value}`: `key = {⟨value⟩}`; `\key-{key}{value}`: `key = ⟨value⟩`

`\keyis*{⟨name⟩}{⟨value⟩}`

A key ⟨*name*⟩ set to value ⟨*value*⟩, the optional star prevents an index entry, `\key{keyis}{value}`: `key = value`.

`\choices{⟨clist of choices⟩}`

A list of choices, `\choices{one,two,three}`: `one|two|three`

`\choicekey{⟨name⟩}{⟨clist of choices⟩}`

A key ⟨*name*⟩ with a list of possible values, `\choicekey{key}{one,two,three}`: `key = one|two|three`

`\boolkey{⟨name⟩}`

A boolean key ⟨*name*⟩ with choices `true` and `false`, `\boolkey{key}`: `key = true|false`

`\default{⟨value⟩}`

Markup for a default choice, `\choices{one,\default{two},three}`: `one|two|three`

5.2. **Versioning Commands, Licensing and Related Stuff**

The commands described in this section are provided by the CNLTX class except where indicated differently. These commands are related to information about the legal stuff of a package and where to find it on th world wide web.

\sinceversion{⟨*version*⟩}

Gives a sidenote like the one on the left.

\changedversion{⟨*version*⟩}

Gives a sidenote like the one on the left.

\newnote*{⟨*cs*⟩}[⟨*num*⟩]{⟨*definition*⟩}

Defines a note like \sinceversion. The star makes the note macro short, ⟨*num*⟩ defines the number of mandatory arguments. Optional arguments are not possible. \sinceversion was defined as follows: \newnote*\sinceversion[1]{Introduced in version~#1}

\newpackagename{⟨*cs*⟩}{⟨*name*⟩}

Define a comand ⟨*cs*⟩ that prints ⟨*name*⟩ formatted like CNLTX.

\newname{⟨*cs*⟩}{⟨*first name*⟩}{⟨*second name*⟩}

Defines ⟨*cs*⟩ to write out the full name and add an index entry sorted by the last name. Also defines a starred variant of ⟨*cs*⟩ that only writes the last name but still adds the full index entry.

\lppl

Typesets "LPPL" and adds a corresponding index entry.

\LPPL

Typesets "LaTeX Project Public License" and adds a the same index entry as \lppl.

\license*[⟨*maintenance status*⟩]   Default: maintained

Typesets 'Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (http://www.latex-project.org/lppl.txt). The software has the status "maintained."'. The un-starred variant adds a \par.

\ctan

Typesets "CTAN" and adds a corresponding index entry.

\CTAN

Typesets "Comprehensive TeX Archive Network" and adds the same index entry as \ctan.

\cnltxacronym{⟨*pdf and sort string*⟩}{⟨*acronym*⟩}

Typesets ⟨*acronym*⟩ with small caps and uses ⟨*pdf and sort string*⟩ as PDF string and for sorting the index entry that is added. This command was used to define \lppl and \ctan. *This is not intended as a replacement for packages like acro [Nie13a] or glossaries[Tal13]! In fact it is a "poor man's" solution that allows me not to require one of those packages.*

\pkg*{⟨*package*⟩}

Format the package name ⟨*package*⟩ and add an index entry. The starred variant adds nothing to the index.

\pkgidx{⟨*package*⟩}

Add an index entry for the package ⟨*package*⟩.

\cls*{⟨*class*⟩}

Format the class name ⟨*class*⟩ and add an index entry. The starred variant adds nothing to the index.

\clsidx{⟨*class*⟩}

Add an index entry for the class ⟨*class*⟩.

\CTANurl[⟨*directory*⟩]{⟨*name*⟩}

Writes a CTAN link like the ones in section 3 on page 4 in the footnotes. The predefined directory is macros/latex/contrib. The link address will be:

http://mirrors.ctan.org/⟨*directory*⟩/⟨*name*⟩/.

\needpackage[⟨*directory*⟩]{⟨*name*⟩}

A wrapper for \pkg{#2}\footnote{\CTANurl[#1]{#2}}

\needclass[⟨*directory*⟩]{⟨*name*⟩}

A wrapper for \cls{#2}\footnote{\CTANurl[#1]{#2}}

```
1 \newpackagename{\foothree}{foo-3}%
2 now \foothree\ looks like \cnltx.
3
4 \newname\carlisle{David}{Carlisle}%
5 \carlisle\ is a well-known member of the \LaTeX\ community.  \carlisle* is
6 the author of many packages such as \pkg*{longtable}.
```

now FOO-3 looks like CNLTX.

David Carlisle is a well-known member of the LATEX community. Carlisle is the author of many packages such as longtable.

5.3. Input Source Code Files

Similar to the environments described in section 6.2 on page 10 CNLTX-EXAMPLE provides a few commands for inputting source code files, formatting and printing the source code and inputting the file directly.

\inputexample[⟨*options*⟩]{⟨*file name*⟩}

The equivalent of the example environment, see section 6.2 on page 10.

`\inputsidebyside[`⟨*options*⟩`]{`⟨*file name*⟩`}`
  The equivalent of the `sidebyside` environment, see section 6.2 on the following page.

`\inputsourcecode[`⟨*options*⟩`]{`⟨*file name*⟩`}`
  The equivalent of the `sourcecode` environment, see section 6.2 on the next page.

`\implementation[`⟨*options*⟩`]{`⟨*file name*⟩`}`
Introduced in   A wrapper for `\lstinputlisting[style=cnltx,#1]{#2}`
version 0.5
       It is possible to define further commands like this:

`\newinputsourcefilecmd[`⟨*option*⟩`]{`⟨*control sequence*⟩`}`
  Defines ⟨*control sequence*⟩ as a new source code input command where ⟨*options*⟩ are preset.

       The existing commands have been defined like this:

```
1 \newinputsourcefilecmd\inputexample
2 \newinputsourcefilecmd[side-by-side]\inputsidebyside
3 \newinputsourcefilecmd[code-only]\inputsourcecode
```

6. Available Environments

6.1. Description Environments

CNLTX-DOC defines some description environments used to describe macros, environments or options.

`\begin{commands}`
  A description-like environment for describing commands. While this environment is a list internally and thus recognizes `\item` own commands are used to describe macros. They are explained in section 7.1 on page 11.

`\begin{options}`
  A description-like environment for describing options. While this environment is a list internally and thus recognizes `\item` own commands are used to describe options. They are explained in section 7.2 on page 12.

`\begin{environments}`
  A description-like environment for describing environments. While this environment is a list internally and thus recognizes `\item` own commands are used to describe environments. They are explained in section 7.3 on page 14.

       These environments are lists all using the same internal `\list`. The setup of this list can be changed via an option:

`list-setup` = {⟨*definitions*⟩}
         Default: `\leftmargin=0pt \labelwidth=2em \labelsep=0pt \itemindent=-1em`
  The setup of the `\list` used by the `commands`, `options` and `environments` environments.

6.2. Source code Environments

CNLTX-EXAMPLE defines the following environments that are used to display source code and possibly the output of the source code, too.

\begin{example}[⟨*options*⟩]
This environment is a formatted verbatim environment that also inputs the output of the inputted code. This environment is described in section 7.4 on page 15.

\begin{sidebyside}[⟨*options*⟩]
This environment is a formatted verbatim environment that also inputs the output of the inputted code. Source and output are printed side-by-side. This environment is described in section 7.4 on page 15.

\begin{sourcecode}[⟨*options*⟩]
This environment is a formatted verbatim environment. This environment is described in section 7.4 on page 15.

Introduced in version 0.2

In each of these environments certain hooks are provided that can be used to add definitions you like:

pre-code = {⟨*definitions*⟩}
⟨*definitions*⟩ are placed before the source code is inserted.

after-code = {⟨*definitions*⟩}
⟨*definitions*⟩ are placed after the source code is inserted.

pre-source = {⟨*definitions*⟩}
⟨*definitions*⟩ are placed before the output of the source code is inserted.

after-source = {⟨*definitions*⟩}
⟨*definitions*⟩ are placed after the output of the source code is inserted.

It is possible to define further environments like this:

\newsourcecodeenv[⟨*option*⟩]{⟨*name*⟩}
Defines ⟨*name*⟩ as a new source code environment where ⟨*options*⟩ are preset.

The existing environments have been defined like this:

```
1  \newsourcecodeenv{example}
2  \newsourcecodeenv[side-by-side]{sidebyside}
3  \newsourcecodeenv[code-only]{sourcecode}
```

7. Usage

7.1. Command Descriptions

Inside of the environment `commands` that was introduced in section 6.1 on page 9 items are input via the following command:

\command*{⟨*name*⟩}[⟨*stuff after*⟩]
This macro formats a control sequence with \cs and puts a line break after it. The optional argument allows printing things directly after the command name and can thus be used for adding arguments.

\Default*!{⟨*code*⟩}

<div style="float:left; width:20%">Changed in version 0.3</div>

This command can be placed after \command or \option in order to give a default definition of a macro or a default value of an option. The definition will then be placed on the same line flush right. The star prevents the insertion of \newline after it. The optional bang adds the information that an option is mandatory, *i.e.*, it has to be set.

\expandable

Introduced in version 0.5

Adds the symbol * to the left of a command in the margin to indicate that the command is expandable. This command should be used *immediately* before \command.

\unexpandable

Introduced in version 0.5

Adds the symbol * to the left of a command in the margin to indicate that the command is not expandable. This command should be used *immediately* before \command.

\expandablesign

Introduced in version 0.5

The macro that holds the sign used by \expandable and \unexpandable.

```
1  \begin{commands}
2    \command{cs}
3      This is about foo bar baz.
4    \command{cs}[\marg{arg}]
5      This one has an argument.
6    \command{cs}[\sarg\oarg{option}]
7      This has a star variant and an optional argument.
8    \command{cs}\Default{foo bar}
9      This one has the default replacement text \code{foo bar}
10   \expandable\command{cs}
11     This macro is expandable.
12 \end{commands}
```

\cs
This is about foo bar baz.

> `\cs{`⟨*arg*⟩`}`
>   This one has an argument.
>
> `\cs*[`⟨*option*⟩`]`
>   This has a star variant and an optional argument.
>
> `\cs`   Default: `foo bar`
>   This one has the default replacement text `foo bar`
>
> * │ `\cs`
>   This macro is expandable.

The `\expandablesign` can of course be redefined to something else you like better. For the sake of completeness there is an option that does exactly this:

`expandable-sign = {`⟨*definition*⟩`}`   Default: `\raise-.5ex\hbox{*}`

  Redefines `\expandablesign` to ⟨*definition*⟩.

7.2. Option Descriptions

The `options` environment knows a few more commands to meet all the different kinds of options.

`\opt*`
  An option. The star prevents an index entry.

`\keyval*-{`⟨*key*⟩`}{`⟨*value*⟩`}`
  A key/value option. The optional star prevents an index entry. The optional `-` strips the braces around ⟨*value*⟩, see the example below.

`\keychoice*{`⟨*key*⟩`}{`⟨*list of choices*⟩`}`
  A key/value option where the value is one of a list of choices. The star prevents an index entry.

`\keybool*{`⟨*name*⟩`}`
  A boolean key, that ist a choice key with choices `true` and `false`. The star prevents an index entry.

`\Default*!{`⟨*code*⟩`}`

  This command can be placed after `\command` or `\option` in order to give a default definition of a macro or a default value of an option. The definition will then be placed on the same line flush right. The star prevents the insertion of `\newline` after it. The optional bang adds the information that an option is mandatory, *i.e.*, it has to be set.

`\Module*!{`⟨*name*⟩`}`

  This command can be placed after `\option` but before `\Default` in order to determine the module the option belongs to. It will be written in the left margin next to the option name. The star prevents the insertion of `\newline` after it. The optional bang *adds* an index entry for the module. This is somehow inconsistent with many of the other commands where an optional

star *prevents* an index entry but it fits to the functionality of \Default which is why this syntax was chosen.

The following demonstrates how the commands would be used to create option descriptions:

```
 1 \begin{options}
 2   \opt{foo}
 3     This makes stuff.  Let's add a few more words so that the line gets
 4     filled and we can see how the output actually looks.
 5   \opt*{foo}\Default{bar}
 6     This makes stuff.  Let's add a few more words so that the line gets
 7     filled and we can see how the output actually looks.
 8   \opt{foo}\Module{bar}
 9     This option belongs to \module*{bar}.  Let's add a few more words so
10     that the line gets filled and we can see how the output actually
11     looks.
12   \opt{foo}\Module{bar}\Default{baz}
13     This option belongs to \module*{bar}.  Let's add a few more words so
14     that the line gets filled and we can see how the output actually
15     looks.
16   \keyval{foo}{bar}\Default
17     This makes stuff.  Let's add a few more words so that the line gets
18     filled and we can see how the output actually looks.
19   \keyval{foo}{bar}\Default!
20     This makes stuff.  Let's add a few more words so that the line gets
21     filled and we can see how the output actually looks.
22   \keyval*{foo}{bar}
23     This makes stuff.  Let's add a few more words so that the line gets
24     filled and we can see how the output actually looks.
25   \keyval-{foo}{bar}
26     This makes stuff.  Let's add a few more words so that the line gets
27     filled and we can see how the output actually looks.
28   \keychoice{foo}{one,two,three}
29     This makes stuff.  Let's add a few more words so that the line gets
30     filled and we can see how the output actually looks.
31   \keybool{foo}
32     This makes stuff.  Let's add a few more words so that the line gets
33     filled and we can see how the output actually looks.
34 \end{options}
```

The code above gives the following output:

foo

This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

foo   Default: bar

This makes stuff. Let's add a few more words so that the line gets filled and we can see how the

output actually looks.

`bar` »`foo`

This option belongs to the module `bar`. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`bar` »`foo`   Default: `baz`

This option belongs to the module `bar`. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`foo` = {⟨*bar*⟩}   (initially empty)

This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`foo` = {⟨*bar*⟩}   (required)

This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`foo` = {⟨*bar*⟩}

This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`foo` = ⟨*bar*⟩

This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`foo` = `one|two|three`

This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`foo` = <u>`true`</u>`|false`

This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

7.3. Environment Descriptions

Environment descriptions are made – unsurprisingly – with the `environments` environment. It knows the command `\environment`:

`\environment*{`⟨*name*⟩`}[`⟨*stuff after*⟩`]`

This macro prints the environment name and puts a line break after it. The optional argument allows printing things directly after the environment name and can thus be used for adding arguments.

```
1 \begin{environments}
2   \environment*{foobar}[\oarg{options}]
3     This is environment \env*{foobar}.  The star prevents it from being
4     added to the index.
5 \end{environments}
```

\begin{foobar}[⟨*options*⟩]
   This is environment foobar. The star prevents it from being added to the index.

7.4. Example Code

Example code can be included through the example environment or the sourcecode environment.

```
1 \begin{example}
2   a \LaTeX\ code example
3 \end{example}
```

This example would give:

```
1 a \LaTeX\ code example
```

   a LaTeX code example

Both environments can be influenced by options:

code-only = <u>true</u>|false   Default: false
   Only typeset the code as code but don't include it afterwards. The code box above is an example for the usage of this option. This option has no effect on the sourcecode environment: this is already what that environment does.

side-by-side = <u>true</u>|false   Default: false
   Typeset source and output side by side. The code is input on the left and the output on the right. Side by side examples are typeset in minipage environments with all consequences that come with them (think of \parindent, page breaks …).

code-left = <u>true</u>|false   Default: true
   If true and the option side-by-side is chosen the source code is printed on the right side else on the left.

code-sep = {⟨*definition*⟩}   Default: \hrulefill
   Code that is inserted between a source code and the corresponding output when printed below each other.

   The same example again, this time using side-by-side (which is the same as using the sidebyside environment):

> ₁ a \LaTeX\ code example   a LᴬTEX code example

   side-by-side and code-left = false:

> a LᴬTEX code example   ₁ a \LaTeX\ code example

   The frame around the examples is done by the mdframed package [Dan13]. It is of course possible to customize it:

add-frame-options = {⟨*mdframed options*⟩}   (initially empty)
   Add options to the predefined ones.

frame-options = {⟨*mdframed options*⟩}
              Default: backgroundcolor=cnltxbg,linecolor=cnltx,roundcorner=5pt
   Overwrite the options with new ones.

   The source code is formatted using the listings package [HM13]. Similar options exist to adapt listings' options that are used for formatting the source code. The predifined style has many options that will not be mentioned here. If you're interested you can find them in cnltx-listings.sty or in section B on page 36.

gobble = ⟨*integer*⟩   Default: 2
   The number of initial characters that is gobbled from each line.

add-cmds = {⟨*list of csnames*⟩}   (initially empty)
   A list of control sequence names that should be recognized as a command sequence in the source code examples and should be formatted accordingly. The control sequence names in this list will also get an index entry when they're used in the source example. This is done internally via \csidx. The option should be used to add the new commands that are defined by the package for which you are writing the manual for.

add-silent-cmds = {⟨*list of csnames*⟩}
   A list of control sequence names that should be recognized as a command sequence in the source code examples and should be formatted accordingly. The control sequence names in this list will *not* get an index entry when they're used in the source example. There already is quite a large but far from comprehensive list of silent commands but many are still missing. This option allows you to extend the list on a per document basis.

`add-listings-options` = {⟨*listings options*⟩}   (initially empty)
> Additional options for the listings [HM13] environments. *This redefines the `cnltx` listings style anf this effects all sourcecode environments!*

`listings-options` = {⟨*listings options*⟩}
> Overwrite existing options with new ones. This can be used to build an own style from scratch. *This redefines the `cnltx` listings style anf this effects all sourcecode environments!*

`sourcecode-options` = {⟨*listings options*⟩}

Introduced in
version 0.4
> These options are added to the listings options of the source code environments without redefing the main style. Hence it can be used to locally add options to a source code environment.

`add-envs` = {⟨*list of environment names*⟩}   (initially empty)
> Like `add-cmds` but for environment names.

`add-silent-envs` = {⟨*list of environment names*⟩}
> Like `add-silent-cmds` but for environment names.

7.5. Example File

Let's say you're documenting a package called `mypackage` that provides the command `\mycommand` and the environment `myenv`. The basic manual setup could then look something like this:

```
1 \documentclass[load-preamble]{cnltx-doc}
2 \usepackage[T1]{fontenc}
3 \usepackage[utf8]{inputenc}
4 \setcnltx{
5   package  = mypackage ,
6   authors  = John Doe ,
7   email    = john@doe.com ,
8   add-cmds = {mycommand} ,
9   add-envs = {myenv}
10 }
11 \begin{document}
12   ...
13 \end{document}
```

8. Formatting Possibilities

One of the goals I wanted to achieve with this package is a consistent look and an easy interface for customization. No font choice and no color choice is fixed. In this section ways to change the formatting are shown.

The formatting of the different commands provided by CNLTX and various other properties can be changed in two ways: either by redefining the internal commands that are used for

the formatting or by setting a corresponding option. Both variants are described in the next subsections.

How the colors should be changed is described in section 12 on page 28.

8.1. Formatting by Redefining Hooks

You can change the formatting by redefining the following commands. They're all defined by the CNLTX-EXAMPLE package except where indicated differently.

\codefont   Default: \ttfamily
  This command is used for all formatting of source code.

\sourceformat   Default: \codefont\small
  Formatting of the listings.

\exampleformat   (initially empty)
  Special formatting of the output of a listing.

\versionnoteformat   Default: \footnotesize\sffamily\RaggedRight

provided by   Formatting of the notes introduced in section 5.2 on page 7.
CNLTX-DOC

\packageformat   Default: \sffamily
  The formatting of package names.

\classformat   Default: \sffamily
  The formatting of class names.

\argumentformat   Default: \normalfont\itshape
  The formatting of \meta{⟨*meta*⟩}.

```
1 \renewcommand*\codefont{\sffamily\bfseries}
2 \code{foo} and \cs*{bar}, option \option{baz}
```

foo and **\bar**, option baz

8.2. Formatting by Setting Options

You can change the formatting of by setting the following options. They're all defined by the CNLTX-EXAMPLE package except where indicated differently.

title-format = {⟨*definition*⟩}   Default: \bfseries\scshape

Introduced in   Formatting of the document title.
version 0.2

caption-font = {⟨*definition*⟩}   Default: `\normalfont\small\sffamily`
This option only has any effect if you use the option `load-preamble`, see section 9.4 on page 21 for details on the option.

caption-label-font = {⟨*definition*⟩}   Default: `\normalfont\small\sffamily\scshape`
This option only has any effect if you use the option `load-preamble`, see section 9.4 on page 21 for details on the option.

code-font = {⟨*definition*⟩}   Default: `\ttfamily`
Used for all formatting of source code.

source-format = {⟨*definition*⟩}   Default: `\codefont\small`
Formatting of the listings.

expl-format = {⟨*definition*⟩}   (initially empty)
Special formatting of the output of a listing.

version-note-format = {⟨*definition*⟩}   Default: `\footnotesize\sffamily\RaggedRight`
Formatting of the notes introduced in section 5.2 on page 7.

<div style="float:left">provided by
CNLTX-DOC</div>

acronym-format = {⟨*definition*⟩}   Default: `\scshape`
Formatting of the acronyms as typeset with `\cnltxacronym`.

<div style="float:left">provided by
CNLTX-TOOLS</div>

pkg-format = {⟨*definition*⟩}   Default: `\sffamily`
The formatting of package names.

cls-format = {⟨*definition*⟩}   Default: `\sffamily`
The formatting of class names.

arg-format = {⟨*definition*⟩}   Default: `\normalfont\itshape`
The formatting of `\meta{`⟨*meta*⟩`}`.

default-format = {⟨*code*⟩}   Default: `\uline`
The formatting of `\default`'s argument. ⟨*code*⟩'s last macro should take one argument.

<div style="float:left">Introduced in
version 0.2</div>

```
1 \setcnltx{code-font=\sffamily\itshape}
2 \code{foo} and \cs*{bar}, option \option{baz}
```

foo and *\bar*, option *baz*

9. Commands, Options and Further Settings Directly Related to the CNLTX-DOC Class

9.1. Using Class Options

The CNLTX-DOC class only knows a few options:

load-preamble = <u>true</u>|false   Default: false
   See section 9.4 on the next page for details.

load-preamble+ = <u>true</u>|false   Default: false
   See section 9.5 on page 23 for details.

add-index = <u>true</u>|false   Default: false
   See section 9.5 on page 23 for details.

babel-options = {⟨*options*⟩}   Default: english
   Options given to the babel[16] package. This option only has an effect if load-preamble = true.

scrartcl = {⟨*options*⟩}   (initially empty)
   Options that are passed to the underlying class scrartcl. *All global options you want to use should be given here.*

9.2. Information on the Described Package or Class

A manual for a package or a class needs some information on the described package like the package name, the version number, the date and so on. This information is given with the following options. They are used to build the title page of the manual.

package = {⟨*package*⟩}
   The name of the package that is described. Either this option or class or name should always be given. This command also defines a command sequence from the package name that formats the package name with color and small caps like CNLTX.

class = {⟨*class*⟩}
   The name of the class that is described. Either this option or package or name should always be given. This command also defines a command sequence from the class name that formats the class name with color and small caps like CNLTX.

name = {⟨*name*⟩}
   The name of the class/package that is described. Either this option or package or class should always be given. This command also defines a command sequence from the class name that formats the class name with color and small caps like CNLTX.

authors = {⟨*author list*⟩}

Changed in
version 0.4   Comma separated list of package/class authors. After each author name you can add an email address by writing it in square brackets: Some Name[some@name.com]. Email addresses specified this way get written as a footnote.

version = {⟨*version number*⟩}
   Version number of the package/class. CNLTX tries to extract the information from the given package or class. This option can be used to set it explicitly.

---

16. on CTAN as babel: http://mirrors.ctan.org/macros/latex/required/babel/

date = {⟨*date*⟩}
  Date of the package/class. CNLTX tries to extract the information from the given package or
  class. This option can be used to set it explicitly.

info = {⟨*package/class info*⟩}
  Information about the package/class. CNLTX tries to extract the information from the given
  package or class. This option can be used to set it explicitly.

subtitle = {⟨*subtitle*⟩}
  A subtitle that is typeset *instead* of the package/class info.

url = {⟨*url*⟩}
  The homepage of the package.

email = {⟨*email*⟩}
  A contact email address.

abstract = {⟨*abstract*⟩}
  An abstract of the package/class/manual. This is text typeset in a box of .75\linewidth.
  Actually it does not have to be text but could be an image or whatever you like.

9.3. Building of the Manuals Title Page

If either the package or class has been given an automatic title page is built using the gathered
information. Figure 1 on the following page roughly sketches which informations is used and
how the different elements are arranged on the title page. The page style of the title page is
plain. Additionally a table of contents is automatically built that is set in two columns. The
automatic building of the title page can be prevented by explicitly setting the following option:

build-title = <u>true</u>|false
  The default state depends on other options given like package. However, setting this option to
  false *after* any of the options described in section 9.2 on the previous page will prevent the
  building of a title page and allows you to design your own.

9.4. Predefined Preamble

It is possible to load a part of my standard preamble automatically by passing an option as class
option.

load-preamble
  Class option that preloads part of my custom preamble.

  Using the option will include the following code:

package name formatted with `\titleformat` and scaled to 1.5 of its size

subtitle

version   date

package information

author names (formatted according to specifications for names as defined with `\newname`)

url when specified via the `url` option

email address when specified via the `email` option

abstract when specified via the `abstract` option
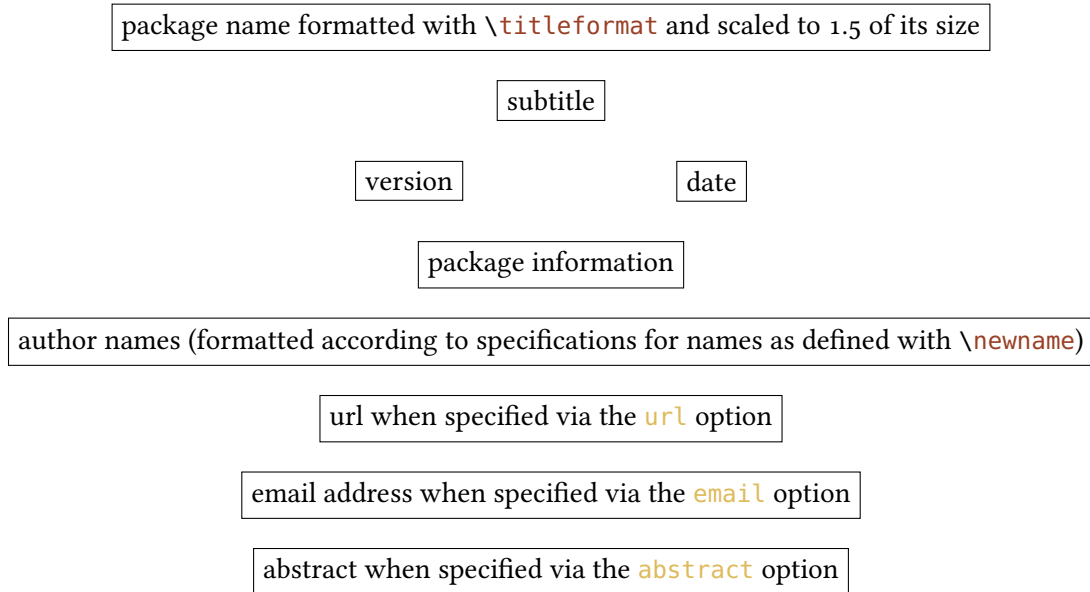
FIGURE 1: Schematic sketch of the title page.

```latex
\RequirePackage{ifxetex,ifluatex}
\ifboolexpr{not bool{xetex} and not bool{luatex}}
  {\RequirePackage[T1]{fontenc}}
  {\RequirePackage{fontspec}}
\RequirePackage[oldstyle]{libertine}
% 'libertinehologopatch' is not on CTAN, yet!
% you can get it at https://bitbucket.org/cgnieder/libertinehologopatch/
\RequirePackage{libertinehologopatch}
\RequirePackage[supstfm=libertinesups]{superiors}
\RequirePackage{microtype}
% activate my microtype settings for tracking of small caps
% with \microtypesetup{tracking=scshape}
% needs files from https://bitbucket.org/cgnieder/microtype-config/
\ifboolexpr{not bool{xetex} and not bool{luatex}}
  {\RequirePackage[scaled=.83]{beramono}}
  {\setmonofont[Scale=MatchLowercase]{Bitstream Vera Sans Mono}}
\RequirePackage{fnpct}
\expandafter\RequirePackage\expandafter[\cnltx@babel@options]{babel}
\renewcommand*\othersectionlevelsformat[3]{%
  \textcolor{cnltx}{#3\autodot}\enskip}
\renewcommand*\partformat{%
  \textcolor{cnltx}{\partname~\thepart\autodot}}
\deffootnote{2em}{1em}{\llap{\thefootnotemark. }}%
\pagestyle{headings}
\setcapindent{1.5em}
```

```
26 \setkomafont{caption}{\cnltx@caption@font}
27 \setkomafont{captionlabel}{\cnltx@captionlabel@font}
```

The effect of this preamble is demonstrated by the document you're reading at this moment.

9.5. Predefined Indexing

CNLTX-DOC allows the automated creation of an index. This is done with the help of the imakeidx package by Enrico Gregorio [Gre13]. To use this feature you have two class options. They cannot be set with `\setcnltx` but must be given as class options.

`add-index` = <u>true</u>|false   Default: `false`
    Enables the automatic creation of an index at the end of the document.

`load-preamble+` = <u>true</u>|false   Default: `false`
    This option has the same effect as adding the options `load-preamble`, `add-index` and `add-bib`.

Enabling the feature

- loads the imakeidx[17] package,

- uses a given style file for the index that can be specified with the `index-style` option,

- sets a certain setup for the index that can be specified with the `index-setup` option and

- adds an index at the end of the document.

The following options are available to customize the appearance of the index:

`index-prologue` = {⟨*text*⟩}
    Adds ⟨*text*⟩ as index prologue between heading and the actual index.

`index-space` = {⟨*dimension*⟩}   Default: `0pt`
    The vertical space between index prologue and index.

`index-setup` = {⟨*options*⟩}   Default: `othercode=\footnotesize,level=\addsec`
    The options that are passed to imakeidx's `\indexsetup` command.

`makeindex-setup` = {⟨*options*⟩}   Default: `columns=2,columnsep=1em`
    The options that are passed to the `\makeindex` command.

`index-style` = {⟨*style file*⟩}   Default: `cnltx.ist`
    The style file that is used for formatting the index.

The index style file `cnltx.ist` contains the following lines:

---

17. on CTAN as imakeidx: http://mirrors.ctan.org/macros/latex/contrib/imakeidx/

```
1 heading_prefix "{\\bfseries "
2 heading_suffix "\\hfil}\\nopagebreak\n"
3 headings_flag  1
4 delim_0 "\\dotfill"
5 delim_1 "\\dotfill"
6 delim_2 "\\dotfill"
7 delim_r "\\nohyperpage{\\textendash}"
8 delim_t ""
9 suffix_2p "\\nohyperpage{\\,\\GetTranslation{cnltx-f.}\\@}"
10 suffix_3p "\\nohyperpage{\\,\\GetTranslation{cnltx-ff.}\\@}"
```

The feature is demonstrated by this document which does not contain a single control sequence containing the string index!

9.6. Bibliography with biblatex

9.6.1. A Bibliography Entry Type package for biblatex

Introduced in
version 0.4

CNLTX-DOC defines a bibliograpy entry type package when biblatex [Leh13] is used. This allows specifying LaTeX packages in bib files:

```
1 @package{pkg:chngcntr,
2   title      = {chngcntr} ,
3   author     = {Peter Wilson} ,
4   maintainer = {Will Robertson} ,
5   date       = {2009-09-02} ,
6   version    = {1.0a} ,
7   url        = {http://mirror.ctan.org/macros/latex/contrib/chngcntr}
8 }
```

As you can see also an entry field maintainer is defined. For this to work you have to use the biblatex bibliography style cnltx. This style basically is a clone of the style alphabetic but defines the necessary additions for the package entry type and the maintainer entry field. Along with the bibliography style a citation style cnltx is provided, again a clone of the alphabetic style. The only addition it makes is that indexing of maintainer names is enabled if biblatex's indexing option is used. *The styles can only be used with the CNLTX-DOC class* as it relies on definitions made by it.

This document uses the following call of biblatex:

```
1 \usepackage[
2   backend=biber,
3   style=cnltx,
4   sortlocale=en_EN,
5   indexing=cite,
```

```
6    useprefix]{biblatex}
7  \addbibresource{cnltx.bib}
```

Actually it let's CNLTX-DOC do it, see section 9.6.2 for details.

Just for the sake of the example I am going to cite the chngcntr package now [Wil09] so you can see both the bibliography entry and the indexed names of package, author and maintainer in the appendix.

9.6.2. Automatic Bibliography

CNLTX-DOC allows the automated creation of a bibliography.

add-bib = <u>true</u>|false   Default: false
    Enables the automatic creation of a bibliography at the end of the document.

load-preamble+ = <u>true</u>|false   Default: false
    This option has the same effect as adding the options load-preamble, add-index and add-bib.

What this options does is including the following code:

```
1  \RequirePackage[
2    backend=biber,
3    style=cnltx,
4    sortlocale=en_EN,
5    indexing=cite,
6    useprefix]{biblatex}
7  \addbibresource{cnltx.bib}
8  \AtEndDocument{\printbibliography}
```

As you can see there's also a bibliography database file cnltx.bib that provides a yet small but growing number of package entries.

10. Predefined listings and mdframed Styles

10.1. mdframed

The source code environments (see section 7.4 on page 15) all get a frame with the help of the mdframed [Dan13] package. For this a custom style is defined called cnltx. The options frame-options and add-frame-options mentioned in section 7.4 on page 15 manipulate this style. It is predefined with these values:

```
1  \def\cnltx@mdframed@options
2    {
3      backgroundcolor = cnltxbg ,
4      linecolor       = cnltx ,
```

```
 5      roundcorner      = 5pt
 6    }
```

10.2. listings

10.2.1. Sourcecode

The code of the source code environments (see section 7.4 on page 15) is formatted with the help of the listings package [HM13]. A listings style is defined called cnltx. The options add-cmds, add-silent-cmds, add-envs, add-silent-envs, listings-options and add-listings-options manipulate this style. It is predefined by CNLTX-EXAMPLE as follows:

```
 1 \def\cnltx@listings@style{
 2    language         = [AlLaTeX]TeX,
 3    alsolanguage     = [plain]TeX,
 4    basicstyle       = {\sourceformat},
 5    numbers          = left,
 6    numberstyle      = \tiny,
 7    xleftmargin      = 1em,
 8    numbersep        = .5em,
 9    gobble           = \cnltx@gobble ,
10    columns          = fullflexible,
11    literate         =
12      {ä}{{\"a}}1
13      {ö}{{\"o}}1
14      {ü}{{\"u}}1
15      {Ä}{{\"A}}1
16      {Ö}{{\"O}}1
17      {Ü}{{\"U}}1
18      {ß}{{\ss}}1 ,
19    breaklines       = true,
20    keepspaces       = true,
21    breakindent      = 1em,
22    commentstyle     = \color{comment},
23    keywordstyle     = \color{cs},
24    deletetexcs      =
25      {
26        a,o,u,A,O,U,
27        begin,
28        center,
29        description,document,
30        end,enumerate,
31        figure,flushleft,flushright,
32        itemize,
33        otherlanguage,
34        table,tabu,tabular
35      },
```

```
36  % \begin, \end:
37  texcsstyle       = [2]\color{beginend},
38  index            = [2][texcs2],
39  indexstyle       = [2]\@gobble,
40  moretexcs        = [2]{begin,end},
41  % environments:
42  texcsstyle       = [3]\color{env},
43  index            = [3][texcs3],
44  indexstyle       = [3]\@gobble,
45  % control sequences:
46  texcsstyle       = [4]\color{cs},
47  index            = [4][texcs4],
48  indexstyle       = [4]\@gobble ,
49  % added control sequences:
50  texcsstyle       = [5]\color{cs},
51  index            = [5][texcs5],
52  indexstyle       = [5]\indexcs,
53  % added environments:
54  texcsstyle       = [6]\color{env},
55  index            = [6][texcs6],
56  indexstyle       = [6]\envidx,
57 }
```

10.2.2. BIBTEX entries

The CNLTX-LISTINGS package defines a listings language BibTeX by defines a huge number of bibentry types and bibentry field types, have a look at section 9.6.1 on page 24. CNLTX-EXAMPLE defines a listings style for formatting them:

```
1  \def\cnltx@bibtex@listings@style{
2  language         = BiBTeX,
3  basicstyle       = {\sourceformat},
4  numbers          = left,
5  numberstyle      = \tiny,
6  xleftmargin      = 1em,
7  numbersep        = .5em,
8  gobble           = \cnltx@gobble ,
9  columns          = fullflexible,
10 literate         =
11   {ä}{{\"a}}1
12   {ö}{{\"o}}1
13   {ü}{{\"u}}1
14   {Ä}{{\"A}}1
15   {Ö}{{\"O}}1
16   {Ü}{{\"U}}1
17   {ß}{{\ss}}1 ,
18 breaklines       = true,
```

```
19    keepspaces      = true,
20    breakindent     = 1em,
21    commentstyle    = \color{comment},
22    keywordstyle    = \color{bibentry} ,
23    keywordstyle    = [2]\color{bibentryfield}\itshape ,
24    showstringspaces = false ,
25 }
```

11. PDF Strings and hyperref

Since the formatting and indexing commands \cs, \env, \option, \pkg, \cls and \key are robust they are ignored in PDF strings. For this reason you should *only use the starred variants* in places where PDF bookmarks are built from such as section titles when you use hyperref [OR12]. Since CNLTX-DOC loads hyperref this means you should do so, too, when you use CNLTX-DOC. This is important for two reasons:

1. Indexing in strings that get written to the table of contents does noch make much sense, anyway, so the starred versions should be used in section titles even if you don't use hyperref.

2. When hyperref is loaded the mentioned commands are disabled in PDF strings in a way that *expects* them to be followed by a star. This means leaving the star out will result in `doesn't match its definition` errors.

12. Predefined Colors and Color-Schemes

12.1. Explicitly Defined Colors

The CNLTX-BASE package defines a number of colors:

cnltxbrown Per default used for the control sequences.

cnltxblue Unused per default.

cnltxred Per default used as base color in various places.

cnltxgreen Unused per default.

cnltxgray Per default used for formatting comments.

cnltxyellow Per default used for options.

cnltxformalblue Unused per default.

cnltxformalred Unused per default.

12.2. Actual Used Color Names and Color Schemes

The colors defined in section 12.1 on the preceding page are not directly used with those names. Instead colors are used whose names describe their function rather than the color. For this the color names are mapped to actual colors and saved as a coloring scheme. There are currently three predefined color schemes whose definitions are given below. Those definitions also show the actually used color names:

The 'default' color scheme is defined as follows:

```
1 \cnltx@define@colorscheme{default}{
2   cs            => cnltxbrown ,  % command sequences
3   option        => cnltxyellow ,% options
4   option        => cnltxyellow ,% modules
5   comment       => cnltxgray ,   % comments
6   beginend      => red ,         % \begin and \end
7   env           => black ,       % environment names
8   argument      => black ,       % argument delimiters
9   meta          => black!80 ,    % arguments of \meta
10  cnltx         => cnltxred ,    % base color
11  cnltxbg       => white ,       % source code box background
12  link          => black!90 ,    % hyperlinks
13  versionnote   => black!75      % versioning notes text
14  bibentry      => cnltxgreen ,  % BibTeX entry types
15  bibentryfield => black,        % BibTeX entry fields
16  expandable    => red ,         % the color used in \expandable
17  unexpandable  => black         % the color used in \unexpandable
18 }
```

The 'blue' color scheme is defined this way:

```
1 \cnltx@define@colorscheme{blue}{
2   cs            => cnltxbrown ,
3   option        => cnltxgreen ,
4   module        => cnltxred ,
5   comment       => cnltxgray ,
6   beginend      => red ,
7   env           => black ,
8   argument      => black ,
9   meta          => black!80 ,
10  cnltx         => cnltxblue ,
11  cnltxbg       => yellow!10 ,
12  link          => cnltx ,
13  versionnote   => black!75
14  bibentry      => cnltxyellow ,
15  bibentryfield => black ,
16  expandable    => red ,
```

```
17    unexpandable  => black
18  }
```

Finally the 'formal' color scheme is defined like this:

```
 1  \cnltx@define@colorscheme{formal}{
 2    cs            => black ,
 3    option        => cnltxformalblue ,
 4    module        => cnltxblue ,
 5    comment       => cnltxgray ,
 6    beginend      => red ,
 7    env           => black ,
 8    argument      => black ,
 9    meta          => black!80 ,
10    cnltx         => cnltxformalblue ,
11    cnltxbg       => white ,
12    link          => black!90 ,
13    versionnote   => black!75 ,
14    bibentry      => black ,
15    bibentryfield => black ,
16    expandable    => red ,
17    unexpandable  => black
18  }
```

13. Language Support

The CNLTX-DOC and the CNLTX-EXAMPLE package both rely on the translations package [Nie13c] for providing some document language dependent strings. Currently only translations for English and German are provided. Others can be added and the existing ones changed with the following command provided by the translations package:

\DeclareTranslation{⟨*language*⟩}{⟨*keyword*⟩}{⟨*translation*⟩}
Provide translations for the string identified by the ID ⟨*keyword*⟩.

The defined strings are listed in table 1 on the next page. They are used in indexing strings and in different parts of the document.

TABLE 1: Overview over available internationalization key words.

Package	key word	English version	German version
CNLTX-EXAMPLE	cnltx-package	package	Paket
CNLTX-EXAMPLE	cnltx-class	class	Klasse
CNLTX-EXAMPLE	cnltx-environment	environment	Umgebung
CNLTX-DOC	cnltx-default	Default	Voreinstellung
CNLTX-DOC	cnltx-empty	initially empty	zunächst leer
CNLTX-DOC	cnltx-required	required	erforderlich
CNLTX-DOC	cnltx-toc	Table of Contents	Inhaltsverzeichnis
CNLTX-DOC	cnltx-license	Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status	Es ist erlaubt, diese Software zu kopieren und zu verteilen unter den Bedingungen der LaTeX Project Public License (LPPL), Version 1.3 oder später. (`http://www.latex-project.org/lppl.txt`). Sie hat den Status
CNLTX-DOC	cnltx-introduced	Introduced in version	Eingeführt in Version
CNLTX-DOC	cnltx-changed	Changed in version	Geändert in Version
CNLTX-DOC	cnltx-f.	f.	f.
CNLTX-DOC	cnltx-ff.	ff.	ff.
CNLTX-DOC	cnltx-maintainer	current maintainer	aktueller Maintainer
CNLTX-DOC	cnltx-maintainer	current maintainers	aktuelle Maintainer

Part III.
Appendix

A. Internal Helper Commands

The commands in this section are only described for the sake of completeness. They are not meant to be used in a document.

A.1. Defined by CNLTX-BASE

Especially CNLTX-BASE defines some useful helper macros that are also used by the other packages and classes.

`\cnltx@@date`
 The creation date of the current version of the bundle.

`\cnltx@@version`
 The version number of the bundle.

`\cnltx@@info`
 The short description of the bundle.

`\cnltx@create@message`∗{⟨*module*⟩}{Error|Warning|WarningNoLine|Info}
Changed in version 0.2
 Create suiting error and warning messaging commands for the module ⟨*module*⟩. The starred version creates messages for a class the un-starred version messages for a package.

`\cnltx@base@error`{⟨*message*⟩}
 Issue an error message using `\PackageError`{cnltx-base}.

`\cnltx@base@warning`{⟨*message*⟩}
 Issue a warning message using `\PackageWarning`{cnltx-base}.

`\cnltx@base@warningnoline`{⟨*message*⟩}
 Issue a warning message using `\PackageWarningNoLine`{cnltx-base}.

`\cnltx@base@info`{⟨*message*⟩}
 Issue a message using `\PackageInfo`{cnltx-base}.

`\cnltx@par`
 Expands to `\par`. Sometimes you need to smuggle a `\par` in a short macro …

`\cnltx@ifsym`{⟨*token*⟩}{⟨*true*⟩}{⟨*false*⟩}
 A generic version of LaTeX's `\@ifstar` that checks if ⟨*token*⟩ follows if the input stream. If yes it is removed and ⟨*true*⟩ is placed in the input stream else ⟨*false*⟩.

`\cnltx@ifdash`{⟨*true*⟩}{⟨*false*⟩}
 A wrapper for `\cnltx@ifsym`{-}.

A. Internal Helper Commands

`\cnltx@ifbang{⟨true⟩}{⟨false⟩}`
   A wrapper for `\cnltx@ifsym{!}`.

`\cnltx@expand@arg{⟨cs⟩}{⟨macro⟩}`
   Expands ⟨*macro*⟩ once before it is passed as argument to ⟨*cs*⟩.

`\cnltx@fullexpand@arg{⟨cs⟩}{⟨argument⟩}`
   Exhaustive expansion of ⟨*argument*⟩ before it is passed as argument to ⟨*cs*⟩.

`\cnltx@fullexpand@twoargs{⟨cs⟩}{⟨argument1⟩}{⟨argument2⟩}`
   Exhaustive expansion of ⟨*argument1*⟩ and ⟨*argument2*⟩ before they're passed as argumenst to ⟨*cs*⟩. This is an alias of the kernel command `\@expandtwoargs` defined for the sake of consistency.

`\cnltx@fullexpand@afterarg{⟨cs⟩}{⟨argument⟩}{⟨code⟩}`

   Places the exhaustivly exanpanded ⟨*code*⟩ after ⟨*cs*⟩{⟨*argument*⟩} in the input stream (*e.g.* for expanding an argument to an environment).

`\cnltx@stripbs`
   A shortcut for `\expandafter\@gobble\string`.

`\cnltx@if@in{⟨tokenlist⟩}{⟨search⟩}{⟨true⟩}{⟨false⟩}`
   Places ⟨*true*⟩ in the input stream if ⟨*search*⟩ is found in ⟨*tokenlist*⟩ and ⟨*false*⟩ if it isn't.

`\cnltx@replace@once{⟨cs⟩}{⟨search⟩}{⟨replace⟩}`
   Replaces the first occurence of ⟨*search*⟩ in the first expansion of ⟨*cs*⟩ with ⟨*replace*⟩.

`\cnltx@long@replace@once{⟨cs⟩}{⟨search⟩}{⟨replace⟩}`

   The same as `\cnltx@replace@once` but ⟨*cs*⟩ will be redefined with `\long`.

`\cnltx@replace@all{⟨cs⟩}{⟨search⟩}{⟨replace⟩}`
   Replaces all occurences of ⟨*search*⟩ in the first expansion of ⟨*cs*⟩ with ⟨*replace*⟩.

`\cnltx@long@replace@all{⟨cs⟩}{⟨search⟩}{⟨replace⟩}`

   The same as `\cnltx@replace@all` but ⟨*cs*⟩ will be redefined with `\long`.

`\cnltx@remove@once{⟨cs⟩}{⟨search⟩}`

   Removes the first occurence of ⟨*search*⟩ in the first expansion of ⟨*cs*⟩.

`\cnltx@long@remove@once{⟨cs⟩}{⟨search⟩}`

   The same as `\cnltx@remove@once` but ⟨*cs*⟩ will be redefined with `\long`.

`\cnltx@remove@all{⟨cs⟩}{⟨search⟩}`

   Removes all occurences of ⟨*search*⟩ in the first expansion of ⟨*cs*⟩.

`\cnltx@long@remove@all{⟨cs⟩}{⟨search⟩}`

   The same as `\cnltx@remove@all` but ⟨*cs*⟩ will be redefined with `\long`.

`\cnltx@define@colorscheme{⟨name⟩}{⟨scheme definition⟩}`
   Command that can be used to define a color scheme.

A.2. Defined by CNLTX-DOC

\cnltx@doc@error{⟨*message*⟩}
  Issue an error message using \ClassError{cnltx-doc}.

\cnltx@doc@warning{⟨*message*⟩}
  Issue a warning message using \ClassWarning{cnltx-doc}.

\cnltx@doc@warningnoline{⟨*message*⟩}
  Issue a warning message using \ClassWarningNoLine{cnltx-doc}.

\cnltx@doc@info{⟨*message*⟩}
  Issue a message using \ClassInfo{cnltx-doc}.

\cnltx@getfileinfo{⟨*file name*⟩}{⟨*file extension*⟩}
  Extract the date, version and background information for a package or a class.

\cnltx@version@note{⟨*note*⟩}
  Command that is used for the versioning notes interally. Sets \reversemarginpar and then writes the note ⟨*note*⟩ to the margin with corresponding formatting.

\begin{cnltxlist}
  The list environment that is used by the environments commands, options and environments.

A.3. Defined by CNLTX-EXAMPLE

\cnltx@example@error{⟨*message*⟩}
  Issue an error message using \PackageError{cnltx-example}.

\cnltx@example@warning{⟨*message*⟩}
  Issue a warning message using \PackageWarning{cnltx-example}.

\cnltx@example@warningnoline{⟨*message*⟩}
  Issue a warning message using \PackageWarningNoLine{cnltx-example}.

\cnltx@example@info{⟨*message*⟩}
  Issue a message using \PackageInfo{cnltx-example}.

\cnltxat
  Robust command that typesets '@' with category code 11. An @ in command names confuses the indexing of the command names. Either one uses another symbol for makeindex's "actual" recognition and also tells idxcmds [Nie13b] about it or one uses \cnltxat in \cs and friends. For the sake of convenience you can define a command like \at that expands to it.[18] In order not to overwrite any such existing macro it is not defined by CNLTX-EXAMPLE. This document for example defines \def\at{\cnltxat}.

---

18. This is important. If you \let it to \cnltxat index entries may be sorted differently! Remember: \cnltxat is robust.

\cnltxletterat
  An alias of \cnltxat.

\cnltxotherat
  The same as \cnltxat but with a '@' with category code 12.

\cnltxbang
  The same as \cnltxotherat except that it contains a '!'.

\cnltxequal
  The same as \cnltxotherat except that it contains a '='.

\cnltx@isvalue
  Used in definitions of the key/value option typesetting commands. Inserts a = with some
  stretchable space around and a legal break-point after it.

\indexcs
  Version of \csidx that takes care of a \textcompwordmark inserted by listings. Also replaces
  all occurences of @ with category code 11 or 12 with \cnltxat. Used to index commands in
  the sourcecode and example environments that have been added with add-cmds.

\newarg[⟨*arg formatting*⟩]{⟨*cs*⟩}{⟨*left delim*⟩}{⟨*right delim*⟩}   Default: \meta

Changed in
version 0.2
  Command used to define the argument commands: \newarg\marg{\{}{\}}. The optional
  argument determines how the argument of the new command will be formatted. This is done
  with \meta per default. \newarg[\code]\Marg{\{}{\}}

\MakePercentComment
  Sets the category code of % to 14.

\cnltx@copyablespace
  Prints a space that is also copyable. Uses the accsupp package by Heiko Oberdiek [Obe01].

\cnltx@mdframed@options
  Predefined option list for the mdframed [Dan13] style cnltx.

\cnltx@listings@style
  Predefined option list for the listings [HM13] style cnltx.

A.4. Defined by CNLTX-LISTINGS

\cnltx@listings@error{⟨*message*⟩}
  Issue an error message using \PackageError{cnltx-listings}.

\cnltx@listings@warning{⟨*message*⟩}
  Issue a warning message using \PackageWarning{cnltx-listings}.

\cnltx@listings@warningnoline{⟨*message*⟩}
  Issue a warning message using \PackageWarningNoLine{cnltx-listings}.

`\cnltx@listings@info{⟨message⟩}`

    Issue a message using `\PackageInfo{cnltx-listings}`.

`\cnltx@predefined@control@sequences`

    A comma-separated list of predefined 'silent' control sequence names.

`\cnltx@predefined@environments`

    A comma-separated list of predefined 'silent' environment names.

`\listsilentcmds`

    Prints all known control sequence names formatted and separated with a comma.

`\listsilentenvs`

    Prints all known environment names formatted and separated with a comma.

A.5. Defined by CNLTX-TOOLS

`\cnltx@tools@error{⟨message⟩}`

    Issue an error message using `\PackageError{cnltx-tools}`.

`\cnltx@tools@warning{⟨message⟩}`

    Issue a warning message using `\PackageWarning{cnltx-tools}`.

`\cnltx@tools@warningnoline{⟨message⟩}`

    Issue a warning message using `\PackageWarningNoLine{cnltx-tools}`.

`\cnltx@tools@info{⟨message⟩}`

    Issue a message using `\PackageInfo{cnltx-tools}`.

`\cnltx@accsupp{⟨actual text⟩}{⟨additional options⟩}{⟨TEX text⟩}`

    A wrapper for package accsupp's `\BeginAccSupp{ActualText = ⟨actual text⟩} ⟨TEX text⟩` `\EndAccSupp{}`.

B. List of Known LATEX Control Sequences

Below are listed all *predefined* control sequence names that are treated as "silent" names by CNLTX, that is, those defined by CNLTX-LISTINGS. You may notice that the list does not cover all control sequences that are formatted. That is because listings [HM13] already has a number of known control sequence names. This list probably overlaps with those on some parts, though.

`\-`, `\@`, `\@alph`, `\@Alph`, `\@arabic`, `\@ctrerr`, `\@empty`, `\@firstofone`, `\@firstoftwo`, `\@gobble`, `\@ifclassloaded`, `\@ifnextchar`, `\@ifpackageloaded`, `\@ifstar`, `\@roman`, `\@Roman`, `\@secondoftwo`, `\@slowromancap`, `\addbibresource`, `\addtokomafont`, `\ae`, `\AE`, `\AfterEndPreamble`, `\AfterPreamble`, `\AfterEndDocument`, `\AfterEndEnvironment`, `\alph`, `\Alph`, `\appto`, `\author`, `\autodot`, `\arabic`, `\AtBeginDocument`, `\AtBeginEnvironment`, `\AtEndDocument`, `\AtEndEnvironment`,

\AtEndPreamble,
\baselineskip,
\BeforeBeginEnvironment,
\begingroup, \bfseries,
\bgroup, \boolfalse,
\booltrue, \c, \caption,
\cb, \centering, \chapter,
\cleardoublepage,
\clearpage, \color, \cref,
\csdef, \cslet, \csletcs,
\csname, \csuse, \d, \date,
\dh, \DH, \DeclareListParser,
\DeclareRobustCommand,
\deffootnote,
\deffootnotemark,
\definecolor,
\descriptionlabel, \dimexpr,
\discretionary, \dj, \DJ,
\documentclass, \egroup,
\emph, \endcsname, \endgroup,
\endnote, \enlargethispage,
\expandonce, \fbox,
\fontsize, \fontshape,
\fontspec, \footnote,
\footnotesize,
\footnotetext, \forlistloop,
\foreignlanguage,
\frenchspacing, \global,
\H, \hskip, \hspace, \huge,
\Huge, \hypersetup,
\hyphenation, \ifblank,
\ifbool, \ifboolexpe,
\ifboolexpr, \ifcsdef,
\ifcsname, \ifdef, \ifinlist,
\ifstr, \ifstrempty,
\ifstrequal, \ignorespaces,
\ignorespacesafterend,
\include, \includeonly,
\indent, \input,
\InputIfFileExists, \item,
\itshape, \j, \k, \KOMAoption,
\KOMAoptions, \l, \L,
\labelenumi, \labelenumii,
\labelenumiii, \labelenumiv,
\label, \labelitemi,
\labelitemii, \labelitemiii,
\labelitemiv, \labelsep,
\large, \Large, \LARGE,
\LaTeX, \LaTeXe, \linebreak,
\linewidth, \listadd,

\LoadClassWithOptions,
\LoadClass, \lstinline,
\lstinputlisting,
\ltx@ifnextchar,
\LuaLaTeX, \LuaTeX,
\maketitle, \mathparagraph,
\mathsection, \mbox,
\mdseries, \NeedsTeXFormat,
\newbool, \newcommand,
\newcounter, \newfontfamily,
\newlabel, \newline,
\newpage, \newrobustcmd,
\ng, \NG, \node, \nolinebreak,
\nonfrenchspacing,
\noindent, \notblank,
\nopagebreak, \normalsize,
\normalfont, \number,
\numexpr, \o, \O, \oe, \OE,
\othersectionlevelsformat,
\P, \pagebreak,
\par, \paragraph,
\parindent, \part,
\partformat, \partname,
\patchcmd, \pdfLaTeX,
\pdfstringdefDisableCommands,
\pdfTeX, \pgfkeys, \pounds,
\preto, \printacronyms,
\printbibliography,
\printendnotes, \printindex,
\ProcessPgfOptions,
\ProcessOptions,
\protected, \protecting,
\providecommand,
\providerobustcmd,
\ProvidesClass,
\ProvidesPackage, \quad,
\qquad, \r, \raggedright,
\raggedleft, \RaggedRight,
\ref, \refstepcounter,
\relax, \renewcommand,
\renewrobustcmd,
\RequirePackage,
\rightarrow, \robustify,
\roman, \Roman,
\rmfamily, \S, \samepage,
\scriptsize, \scshape,
\section, \selectfont,
\selectlanguage,
\setcapindent, \setcounter,
\setfnpct, \setkomafont,

\setlength, \setmainfont,
\setmainlanguage,
\setmonofont,
\setotherlanguage,
\setotherlanguages,
\setsansfont, \shorthandoff,
\shorthandon, \sidenote,
\sffamily, \slshape, \small,
\ss, \SS, \stepcounter,
\subparagraph, \subsection,
\subsubsection, \t,
\tableofcontents, \TeX,
\test, \textasciicircum,
\textasciitilde,
\textasteriskcentered,
\textbackslash, \textbar,
\textbf, \textbraceleft,
\textbraceright, \textcolor,
\textcompwordmark,
\textdollar, \textemdash,
\textendash, \textenglish,
\textexclamdown,
\textgreater, \textit,
\textless, \textmd,
\textogonekcentered,
\textrm, \textsc, \textsf,
\textquestiondown,
\textquotedbl,
\textquotedblleft,
\textquotedblright,
\textquoteleft,
\textquoteright,
\textsc, \textsection,
\textsl, \textsubscript,
\textsuperscript,
\textsterling, \texttt,
\textunderscore, \textup,
\textwidth, \th, \TH,
\the, \theendnotes,
\theenumi, \theenumii,
\theenumiii, \theenumiv,
\thefootnotemark, \thepart,
\tikz, \tiny, \title, \today,
\ttfamily, \two@digits,
\usecounter, \usepackage,
\upshape, \v, \vskip, \vspace,
\xdefinecolor, \XeLaTeX,
\XeTeX

C. List of Known LATEX Environments

Below are listed all *predefined* control sequence names that are treated as "silent" names by CNLTX, that is, those defined by CNLTX-LISTINGS.

center, description, document, enumerate, figure, flushleft,

flushright, itemize, labeling, longtable, otherlanguage, tabbing,

table, tabu, tabular, tabularx, tabulary, verbatim

D. Bibliography

[Ars11] Donald Arseneau. ulem, Mar. 18, 2011.
         URL: http://mirror.ctan.org/macros/latex/contrib/ulem.

[Dan13] Marco Daniel. mdframed. version 1.9b, July 1, 2013.
         URL: http://mirror.ctan.org/macros/latex/contrib/mdframed.

[Gre13] Enrico Gregorio. imakeidx. version 1.3a, July 11, 2013.
         URL: http://mirror.ctan.org/macros/latex/contrib/imakeidx.

[HM13]   Carsten Heinz and Brooks Moses, current maintainer: Jobst Hoffmann.
         listings. version 1.5b, Aug. 26, 2013.
         URL: http://mirror.ctan.org/macros/latex/contrib/listings.

[Ker07] Uwe Kern. xcolor. version 2.11, Jan. 21, 2007.
         URL: http://mirror.ctan.org/macros/latex/contrib/xcolor.

[KN12]   Markus Kohm and Frank Neukahm. KOMA-Script. version 3.11b, July 29, 2012.
         URL: http://mirror.ctan.org/macros/latex/contrib/koma-script.

[Koh12] Markus Kohm. marginnote. version 1.1i, Mar. 29, 2012.
         URL: http://mirror.ctan.org/macros/latex/contrib/marginnote.

[Leh11] Philipp Lehman. etoolbox. version 2.1, Jan. 3, 2011.
         URL: http://mirror.ctan.org/macros/latex/contrib/etoolbox.

[Leh13] Philipp Lehman, current maintainers: Joseph Wright, Audrey Boruvka, and
         Philip Kime. biblatex. version 2.7a, July 14, 2013.
         URL: http://mirror.ctan.org/macros/latex/contrib/biblatex.

[Mit11] Frank Mittelbach. multicol. version 1.7a, June 27, 2011.
         URL: http://mirror.ctan.org/macros/latex/required/tools/multicol.

[Nie13a] Clemens Niederberger. acro. version 1.4a, Sept. 2, 2013.
         URL: http://mirror.ctan.org/macros/latex/contrib/acro.

[Nie13b] Clemens Niederberger. idxcmds. version 0.2b, Aug. 31, 2013.
         URL: http://mirror.ctan.org/macros/latex/contrib/idxcmds.

[Nie13c] Clemens Niederberger. translations. version 1.1, Aug. 5, 2013.
         URL: http://mirror.ctan.org/macros/latex/contrib/translations.

D. Bibliography

[Obe01]   Heiko Oberdiek. accsupp. version 0.3, Jan. 16, 2001.
          URL: http://mirror.ctan.org/macros/latex/contrib/oberdiek/accsupp.

[OR12]   Heiko Oberdiek and Sebastian Rahtz. hyperref. version 6.83m, Nov. 6, 2012.
          URL: http://mirror.ctan.org/macros/latex/contrib/hyperref.

[Rob09]   Will Robertson. trimspaces. version 1.1, Sept. 17, 2009.
          URL: http://mirror.ctan.org/macros/latex/contrib/trimspaces.

[Sch09]   Martin Schröder. ragged2e. version 2.1, May 21, 2009.
          URL: http://mirror.ctan.org/macros/latex/contrib/ms/ragged2e.

[Tal13]   Nicola L.C. Talbot. glossaries. version 3.05, Apr. 21, 2013.
          URL: http://mirror.ctan.org/macros/latex/contrib/glossaries.

[Wil09]   Peter Wilson, current maintainer: Will Robertson.
          chngcntr. version 1.0a, Sept. 2, 2009.
          URL: http://mirror.ctan.org/macros/latex/contrib/chngcntr.

[Wri11]   Joseph Wright. pgfopts. version 2.1, June 2, 2011.
          URL: http://mirror.ctan.org/macros/latex/contrib/pgfopts.

Index