The csquotes package

Context sensitive quotation facilities

Philipp Lehman
plehman@gmx.net

Version 4.1
April 11, 2008

Contents

1 Introduction 1
2 Package options 2
3 Basic interface 3
4 Active quotes 6
5 Integrated interface 8
6 Display environments 9
7 Configuration 11
8 Hints and caveats 18
9 Author interface 21
10 Revision history 25

List of Tables

1 Package options 2
2 Styles and variants 11
3 Language aliases 12
4 Configurable parameters . . 15
5 Auxiliary hooks 16
6 Quotation marks 18

1 Introduction

This document is a systematic reference manual for the `csquotes` package. It is supplemented by a hands-on tutorial including practical examples.[1]

1.1 About

This package provides advanced facilities for inline and display quotations. It is designed for a wide range of tasks ranging from the most simple applications to the more complex demands of formal quotations. The facilities include commands, environments, and user-definable 'smart quotes' which dynamically adjust to their context. Quotation marks are switched automatically if quotations are nested and can adjust to the current language. There are additional facilities designed to cope with the more specific demands of academic writing, especially in the humanities and the social sciences. All quote styles as well as the optional active quotes are freely configurable.

1.2 Requirements

This package requires e-TeX and the `etoolbox` package which is readily available from CTAN.[2] TeX distributions have been shipping e-TeX binaries for quite some time, most distributions even use them by default these days. The `csquotes` package checks if it is running under e-TeX. Simply try compiling your documents as you usually do, the chances are that it just works. If you get an error message, try compiling the document with `elatex` instead of `latex` or `pdfelatex` instead of `pdflatex`, respectively.

1 http://www.ctan.org/tex-archive/macros/latex/contrib/csquotes/tutorial.tex
2 http://www.ctan.org/tex-archive/macros/latex/contrib/etoolbox/

Option key	Possible values
strict	true, false
babel	true, false, try, once, tryonce
style	⟨*style*⟩ or ⟨*alias*⟩
danish	quotes, guillemets
english	american, british
french	quotes, quotes*, guillemets, guillemets*
german	quotes, guillemets, swiss
italian	quotes, guillemets
norwegian	guillemets, quotes
spanish	spanish, mexican
swedish	quotes, guillemets, guillemets*

Table 1: Package options defined by default

1.3 License

1.4 Contributions

The multilingual support of this package depends on user contributions. If you want to contribute a quote style, please refer to section 7.1 for an overview of the components a quote style is composed of and to table 6 for a list of commands which should be used in the definition of quote styles.

1.5 Acknowledgments

I am indebted to Donald Arseneau, Mark Wooding, and David Carlisle for valuable hints in the early stages of development. Frank Mittelbach kindly provided hooks in the `inputenc` package. Additional thanks go to Robert Schlicht.

2 Package options

Package options are given in ⟨*key*⟩=⟨*value*⟩ syntax. Table 1 indicates the default option keys and their possible values. Additional options may be defined in the configuration file. See sections 7.3 and 7.5 for details.

2.1 The `strict` option

Turns all package warnings into error messages. This is useful to ensure that no problem will go unnoticed when finalizing a document. The short form `strict` is equivalent to `strict=true`.

2.2 The `babel` option

Controls multilingual support. `babel=true` continuously adapts the quote style to the language of the `babel` package. `babel=once` will only adapt the style once so that it matches `babel`'s main language. `babel=try` and `babel=tryonce` are similar

1 http://www.ctan.org/tex-archive/macros/latex/base/lppl.txt

to babel=true and babel=once if babel is loaded but will not issue any warnings if it is not. The short form babel is equivalent to babel=true. Multilingual support is not enabled by default. See also section 3.7.

2.3 The style option

Selects a fixed quotation style. The style is used throughout the document unless it is changed manually, see section 3.7 for details. This option implicitly sets babel=false. Please refer to tables 2 and 3 for a list of available quote styles and aliases. See sections 7.1, 7.2, and 7.5 for instructions on adding or modifying quote styles.

2.4 The language options

Use the language options listed in table 1 to choose a style variant if there is more than one. The first variant in the list is the default for the respective style. In the following example, the quote style would generally be adapted to the current language using the default style for that language. In the English parts of the text, the quotation marks would follow the British standard. The German parts would use guillemets instead of curly quotes:

```
\usepackage[english,ngerman]{babel}
\usepackage[babel,english=british,german=guillemets]{csquotes}
```

Note that babel's language name is ngerman here whereas this package uses german. When selecting a quote style automatically, this package will essentially normalize the language names first, using a list of aliases which map languages to quote styles. See section 7.5 and table 3 for details. See section 8.9 for some additional notes concerning the predefined styles.

3 Basic interface

This package supports two ways to tag quotations: built-in commands and active quotes defined in the document preamble or the configuration file. This section will introduce the basic commands, active quotes are discussed in section 4. When working with automated citations, you might also want to learn about the integrated quotation facilities presented in section 5.

3.1 Quoting regular text

The most basic command will simply enclose its argument in quotation marks:

\enquote{⟨*text*⟩}
\enquote*{⟨*text*⟩}

Like all quotation facilities, this command is context sensitive. Depending on the nesting level, it will toggle between outer and inner quotation marks with plain and nested quotations. The starred version of this command skips directly to the inner level. If multilingual support is enabled, the style of all quotation marks will be adapted to the current language.

3.2 Quoting text in a foreign language

To facilitate typesetting quotations in a foreign language, there are two commands which combine `\enquote` with `babel`'s language switches:

`\foreignquote{`⟨*lang*⟩`}{`⟨*text*⟩`}`
`\foreignquote*{`⟨*lang*⟩`}{`⟨*text*⟩`}`

This command combines `\enquote` with `\foreignlanguage`. It switches hyphenation patterns and enables the extra definitions provided by the `babel` package for ⟨*lang*⟩, which must be a language name known to `babel`. The quotation marks will match the language of the quoted piece of text.

`\hyphenquote{`⟨*lang*⟩`}{`⟨*text*⟩`}`
`\hyphenquote*{`⟨*lang*⟩`}{`⟨*text*⟩`}`

This command combines `\enquote` with the `hyphenrules` environment, that is, it merely switches hyphenation patterns. The quotation marks will match the language of the text surrounding the quotation.

3.3 Formal quoting of regular text

Formal quotations are always accompanied by a citation indicating the source of the quoted text. The following command is an extended version of `\enquote` which encloses the quoted piece of text in quotation marks and adds a citation after the quotation:

`\textquote[`⟨*cite*⟩`][`⟨*punct*⟩`]{`⟨*text*⟩`}`
`\textquote*[`⟨*cite*⟩`][`⟨*punct*⟩`]{`⟨*text*⟩`}`

The argument ⟨*text*⟩ may be any arbitrary piece of text to be enclosed in quotation marks. The optional arguments ⟨*cite*⟩ and ⟨*punct*⟩ specify the citation and any terminal punctuation which is not part of ⟨*text*⟩. See section 7.8 on how to change the way these arguments are handled. The starred version of this command skips directly to the inner quotation level. Here are some usage examples:

```
\textquote{...}
\textquote[][?]{...}
\textquote[Doe 1990, 67]{...}
\textquote[{\cite[67]{doe90}}]{...}
```

Note the use of the optional arguments in the examples above. As shown in the second example, ⟨*cite*⟩ has to be given whenever ⟨*punct*⟩ is used, even if it is empty. Also keep in mind that an optional argument containing square brackets must be wrapped in an additional pair of curly braces as shown in the last example. When working with automated citations, you might also want to learn about the integrated quotation facilities presented in section 5.

3.4 Formal quoting of text in a foreign language

There are two additional commands which combine `\textquote` with `babel`'s language switches:

`\foreigntextquote{`⟨*lang*⟩`}[`⟨*cite*⟩`][`⟨*punct*⟩`]{`⟨*text*⟩`}`
`\foreigntextquote*{`⟨*lang*⟩`}[`⟨*cite*⟩`][`⟨*punct*⟩`]{`⟨*text*⟩`}`

This command combines `\textquote` with `\foreignlanguage`. Apart from the language, the arguments are handled as with `\textquote`.

`\hyphentextquote{`⟨*lang*⟩`}[`⟨*cite*⟩`][`⟨*punct*⟩`]{`⟨*text*⟩`}`
`\hyphentextquote*{`⟨*lang*⟩`}[`⟨*cite*⟩`][`⟨*punct*⟩`]{`⟨*text*⟩`}`

This command combines `\textquote` with the `hyphenrules` environment. Apart from the language, the arguments are handled as with `\textquote`.

3.5 Block quoting of regular text

A common requirement in academic writing demands that quotations be embedded in the text if they are short but set off as an indented paragraph if they are longer than a certain number of lines. In the latter case no quotation marks are inserted. The following command deals with this requirement automatically:

`\blockquote[`⟨*cite*⟩`][`⟨*punct*⟩`]{`⟨*text*⟩`}`

This command determines the number of lines required to typeset ⟨*text*⟩. If ⟨*text*⟩ is longer than a given number of lines or if it spans more than one paragraph, it is wrapped in a block quotation environment. Otherwise `\blockquote` behaves like `\textquote`. Quotations in footnotes, parboxes, minipages, and floats are always set inline. By default, the threshold is three lines and the environment used for long quotations is the `quote` environment. See section 7.7 on how to change these parameters. The optional arguments ⟨*cite*⟩ and ⟨*punct*⟩ specify the citation and any terminal punctuation which is not part of ⟨*text*⟩. See section 7.8 on how to change the way these arguments are handled.

3.6 Block quoting of text in a foreign language

There are two additional commands which combine `\blockquote` with `babel`'s language switches:

`\foreignblockquote{`⟨*lang*⟩`}[`⟨*cite*⟩`][`⟨*punct*⟩`]{`⟨*text*⟩`}`

This command behaves the same as `\foreignquote` if the quotation is short. If it exceeds the threshold or spans several paragraphs, it will be wrapped in an `otherlanguage*` environment which is in turn wrapped in a block quotation environment. The arguments are handled as with `\blockquote`.

`\hyphenblockquote{`⟨*lang*⟩`}[`⟨*cite*⟩`][`⟨*punct*⟩`]{`⟨*text*⟩`}`

This command works like `\hyphenquote` if the quotation is short. If it exceeds the threshold or spans several paragraphs, it will be wrapped in an `hyphenrules` environment which is in turn wrapped in a block quotation environment. The arguments are handled as with `\blockquote`.

3.7 Selecting quote styles

Quote styles may be selected manually at any point in the document body by way of the following command:

`\setquotestyle[`⟨*variant*⟩`]{`⟨*style*⟩`}`
`\setquotestyle{`⟨*alias*⟩`}`
`\setquotestyle*`

The regular form of this command selects a quote style and disables multilingual support. Its mandatory argument may be a quote style or an alias. If it is a quote style, the optional argument indicates the style variant. The starred version, which takes no arguments, enables multilingual support. Please refer to tables 2 and 3 for a list of available styles, style variants, and language aliases.

4 Active quotes

This package also supports active characters corresponding to the commands presented in section 3. Roughly speaking, an active character is a single character which functions as a command. Like the corresponding control sequences, active quotes are fully-fledged markup elements which verify the nesting level and issue an error if quotations are nested in an invalid way. If multilingual support is enabled, the style of all quotation marks will be adapted to the current language. The commands presented in the following allocate such active quotes. They may be used in the configuration file, the preamble, or the document body. Note that all characters are automatically checked for validity as they are allocated. This package will reject characters which are unsuitable as active quotes. See section 8.3 for details on the characters which may be used as active quotes.

4.1 Quoting regular text

`\MakeOuterQuote` and `\MakeInnerQuote` define active quotes which print outer and inner quotation marks. Both take one mandatory argument, the character serving as both opening and closing mark:

`\MakeOuterQuote{`⟨*character*⟩`}`
`\MakeInnerQuote{`⟨*character*⟩`}`

`\MakeAutoQuote` defines active quotes which toggle between outer and inner quotations automatically. The two mandatory arguments serve as opening and closing mark and must be distinct:

`\MakeAutoQuote{`⟨*character 1*⟩`}{`⟨*character 2*⟩`}`
`\MakeAutoQuote*{`⟨*character 1*⟩`}{`⟨*character 2*⟩`}`

All active quotes defined with `\MakeAutoQuote` work like `\enquote`. Those defined with `\MakeOuterQuote` and `\MakeInnerQuote` cover only a part of this functionality. The former correspond to the outer level of `\enquote` whereas the latter correspond to the starred version. `\MakeAutoQuote*` is similar to `\MakeInnerQuote`, i.e. it corresponds to `\enquote*`.

4.2 Quoting text in a foreign language

These commands combine `\MakeAutoQuote` with `babel`'s language switches:

`\MakeForeignQuote`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*character 2*⟩}
`\MakeForeignQuote*`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*character 2*⟩}
`\MakeHyphenQuote`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*character 2*⟩}
`\MakeHyphenQuote*`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*character 2*⟩}

The active quotes defined with the above commands are similar in function to `\foreignquote` and `\hyphenquote`, respectively. The starred variants correspond to `\foreignquote*` and `\hyphenquote*`.

4.3 Block quoting of regular text

`\MakeBlockQuote` defines active quotes which will set quotations inline or as a separate paragraph, depending on their length. This command takes three mandatory arguments which must be distinct:

`\MakeBlockQuote`{⟨*character 1*⟩}{⟨*delimiter*⟩}{⟨*character 2*⟩}

The arguments are automatically checked for validity, see section 8.3 for details. All active quotes defined with `\MakeBlockQuote` behave essentially the same as `\blockquote`, but the handling of the citation is slightly different. ⟨*character 1*⟩ will serve as the opening mark in the source file, ⟨*character 2*⟩ as the closing one. The character indicated by the middle argument ⟨*delimiter*⟩ will serve as a delimiter separating the quoted text from the citation which is given last as the active quotes are used:

```
\MakeBlockQuote{<}{|}{>}
...
<text|citation>
```

If the delimiter is omitted, the entire text between the opening and the closing mark will be treated as quotation text. See the tutorial for more usage examples.

4.4 Block quoting of text in a foreign language

These commands combine `\MakeBlockQuote` with `babel`'s language switches:

`\MakeForeignBlockQuote`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*delimiter*⟩}{⟨*character 2*⟩}
`\MakeHyphenBlockQuote`{⟨*lang*⟩}{⟨*character 1*⟩}{⟨*delimiter*⟩}{⟨*character 2*⟩}

The active quotes defined with `\MakeForeignBlockQuote` behave essentially the same as `\foreignblockquote`. Those defined with `\MakeHyphenBlockQuote` work like `\hyphenblockquote`. The behavior of the delimiter character is similar to `\MakeBlockQuote`.

4.5 Controlling active quotes

The commands introduced above merely allocate active quotes, but these characters are not immediately made active. All allocated quotes are automatically enabled at the beginning of the document body. If any active quotes are allocated in

the document body, they need to be enabled with `\EnableQuotes`. The following commands control the state of the active quotes within a local scope.

`\EnableQuotes` Enables all active quotes by redefining the allocated characters and making them active. It also restores them when disabled, set to verbatim, or overwritten.

`\DisableQuotes` Disables all active quotes by restoring the original category codes and definitions of all allocated characters.

`\VerbatimQuotes` Switches to verbatim active quotes. All active quotes will be printed verbatim until their default behavior is restored with `\EnableQuotes`.

`\DeleteQuotes` Disables and deallocates all active quotes, i.e. performs a complete reset of all allocated characters so that they may be newly defined.

5 Integrated interface

The commands presented in this section are extended versions of some of those discussed in section 3. They differ from their counterparts in that they integrate automated citations into their syntax. Instead of adding `\cite` manually, you pass the citation arguments to the respective quotation command. See section 7.7 on how to use a command other than `\cite` to handle the citations.

5.1 Formal quoting of regular text

The most basic integrated command is an extended version of `\textquote`:

`\textcquote`[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}
`\textcquote*`[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}

⟨*text*⟩ may be any arbitrary piece of text. The optional argument ⟨*punct*⟩ specifies any terminal punctuation which is not part of ⟨*text*⟩. See section 7.8 on how to change the way this argument is handled. The starred version of this command skips directly to the inner quotation level. The remaining arguments are handed over to `\cite`. Note that `\cite` normally supports one optional argument only. ⟨*prenote*⟩ is only available in conjunction with the `natbib`, `jurabib`, and `biblatex` packages. How these arguments are handled depends on the citation command. With `natbib` and `biblatex`, ⟨*prenote*⟩ is in fact a notice such as 'see'. With jurabib, this argument has a different function by default. The argument ⟨*postnote*⟩, which is always available, indicates the citation postnote. This is usually a page number. ⟨*key*⟩ is the citation key. See sections 7.7 and 7.8 on how to customize the citation. Also see the tutorial for usage examples.

5.2 Formal quoting of text in a foreign language

These commands combine `\textcquote` with `babel`'s language switches:

`\foreigntextcquote`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}
`\foreigntextcquote*`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}

This command combines `\textcquote` with `\foreignlanguage`. The handling of the arguments is similar to `\textcquote`.

`\hyphentextcquote`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}
`\hyphentextcquote*`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}

This command combines `\textcquote` with the `hyphenrules` environment. The handling of the arguments is similar to `\textcquote`.

5.3 Block quoting of regular text

Block quotations may be combined with automated citations as well. The core of the integrated block quotation facilities is an extended version of `\blockquote`:

`\blockcquote`[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}

The difference between `\blockcquote` and `\blockquote` is that there are three citation arguments instead of one. The handling of these citation arguments is similar to `\textcquote`; see section 5.1 for details. Also see sections 7.7 and 7.8 on how to customize block quotations.

5.4 Block quoting of text in a foreign language

These commands combine `\blockcquote` with `babel`'s language switches:

`\foreignblockcquote`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}

This command combines `\blockcquote` with `\foreignlanguage`. Long quotations will be wrapped in an `otherlanguage*` environment. The handling of the citation arguments is similar to `\textcquote`.

`\hyphenblockcquote`{⟨*lang*⟩}[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*key*⟩}[⟨*punct*⟩]{⟨*text*⟩}

This command combines `\blockcquote` with the `hyphenrules` environment. The handling of the citation arguments is similar to `\textcquote`.

6 Display environments

The environments introduced in this section will typeset quotations as a separate paragraph which looks exactly like a long quotation set by means of the block quotation facilities. Use them for quotations which are to be presented as a separate paragraph regardless of their length. Note that these environments are not replacements for the standard `quote` environment in the strict sense. They function as an additional layer on top of the latter, just like the block quotation facilities. The advantage of using these environments instead of resorting to the standard `quote` environment is that they are configurable, support citations, and will always be in sync with the block quotation facilities with respect to the configuration options discussed in sections 7.7 and 7.8.

6.1 Basic display environments

The arguments of all display environments are generally appended to the `\begin` section of the environment:

```
\begin{displayquote} [⟨cite⟩][⟨punct⟩]
  \end{displayquote}
```

The two optional arguments of this environment specify the citation and any terminal punctuation which is not part of the quoted text. The citation will be inserted at the end of the environment. Trailing horizontal white space at the end of the environment is removed automatically. See sections 7.7 and 7.8 on how to customize the display environments. There are two additional environments which combine `displayquote` with `babel`'s language switches:

```
\begin{foreigndisplayquote} {⟨lang⟩}[⟨cite⟩][⟨punct⟩]
  \end{foreigndisplayquote}
```

This environment combines `displayquote` with `otherlanguage*`. Apart from the language, the arguments are handled as with `displayquote`.

```
\begin{hyphendisplayquote} {⟨lang⟩}[⟨cite⟩][⟨punct⟩]
  \end{hyphendisplayquote}
```

This environment combines `displayquote` with `hyphenrules`. Apart from the language, the arguments are handled as with `displayquote`.

6.2 Integrated display environments

The following environment is an extended version of `displayquote`:

```
\begin{displaycquote} [⟨prenote⟩][⟨postnote⟩]{⟨key⟩}[⟨punct⟩]
  \end{displaycquote}
```

The difference is that there are three citation arguments instead of one in this case. The placement of the citation is similar to `displayquote`. The citation arguments are handled as with `\textcquote`; see section 5.1 for details. Also see sections 7.7 and 7.8 on how to customize this environment. There are two environments which combine `displaycquote` with `babel`'s language switches:

```
\begin{foreigndisplaycquote} {⟨lang⟩}[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}[⟨punct⟩]
  \end{foreigndisplaycquote}
```

This environment combines `displaycquote` with `otherlanguage*`. Apart from the language, the arguments are handled as with `displaycquote`.

```
\begin{hyphendisplaycquote} {⟨lang⟩}[⟨prenote⟩][⟨postnote⟩]{⟨key⟩}[⟨punct⟩]
  \end{hyphendisplaycquote}
```

This environment combines `displaycquote` with `hyphenrules`. Apart from the language, the arguments are handled as with `displaycquote`.

Quote style	Style variants
danish	quotes, guillemets
dutch	–
english	american, british
finnish	–
french	quotes, quotes*, guillemets, guillemets*
german	quotes, guillemets, swiss
italian	quotes, guillemets
norwegian	guillemets, quotes
spanish	spanish, mexican
swedish	quotes, guillemets, guillemets*

Table 2: Quote styles and style variants defined by default

7 Configuration

If available, this package will load the configuration file `csquotes.cfg`. You may use this file to define new quote styles and aliases or redefine existing ones.

7.1 Defining quote styles

The following command defines additional quote styles and variants or redefines existing ones:

`\DeclareQuoteStyle`[⟨*variant*⟩]{⟨*style*⟩}[⟨*outer init*⟩][⟨*inner init*⟩]%
{⟨*opening outer mark*⟩}[⟨*middle outer mark*⟩]{⟨*closing outer mark*⟩}[⟨*kern*⟩]%
{⟨*opening inner mark*⟩}[⟨*middle inner mark*⟩]{⟨*closing inner mark*⟩}

This command may be used in the configuration file or in the document preamble. The term 'outer' refers to the first quotation level, 'inner' means quotations within another quotation. A 'middle mark' is a quotation mark inserted at the beginning of every paragraph within a quotation spanning multiple paragraphs. In most cases, the arguments defining the quotation marks will simply contain one of the commands listed in table 6. If both an outer and an inner quotation begin or end simultaneously, the kerning specified by the value ⟨*kern*⟩ will be inserted between the adjoining quotation marks. While this value can be given in any unit known to TeX, it is advisable to use the relative, font-dependent unit 'em' instead of absolute units such as points, inches, or millimeters. Note that ⟨*kern*⟩ is used as a fallback value only. If the font provides kerning data for the respective pair of quotation marks the font's kerning takes precedence.
⟨*outer init*⟩ and ⟨*inner init*⟩ are all-purpose hooks initializing the quote style. Selecting a quote style will make these hooks available to all quotation commands without expanding them. The execution of ⟨*outer init*⟩ will take place immediately before the opening outer quote is inserted, but inside the group formed by the quotation. ⟨*inner init*⟩ is executed before the opening inner quote is inserted. It is advisable to avoid any global assignments in this context to prevent interference with other styles. Whenever ⟨*inner init*⟩ is used ⟨*outer init*⟩ has to be given as well, even if the argument is empty. Refer to table 2 for a list of all predefined quote styles and their variants. These are the backend styles only, see also table 3 for a

Alias	Backend style or alias	Alias	Backend style or alias
american	english/american	naustrian	austrian
australian	english/british	newzealand	english/british
austrian	german/quotes	ngerman	german
british	english/british	norsk	norwegian
canadian	english/american	norwegian	norwegian/guillemets
danish	danish/quotes	nynorsk	norwegian
english	english/american	spanish	spanish/spanish
french	french/quotes	swedish	swedish/quotes
german	german/quotes	swiss	german/swiss
italian	italian/quotes	UKenglish	british
mexican	spanish/mexican	USenglish	american

Table 3: Language aliases defined by default

list of language aliases. See section 7.5 for some examples as well as an illustration of how quote styles, aliases, and package options interact.

7.2 Defining quote aliases

The following command defines quote aliases:

`\DeclareQuoteAlias`[⟨*variant*⟩]{⟨*style*⟩}{⟨*alias*⟩}
`\DeclareQuoteAlias`{⟨*first-level alias*⟩}{⟨*second-level alias*⟩}

This command may be used in the configuration file or in the document preamble. The alias may point to a backend style or to another alias. Most language aliases refer to a backend style, but some point to an intermediate alias instead. If the alias is defined for the sake of the `babel` package, its name must be identical to the language name used by `babel`, i.e. the expansion of `\languagename`. See section 7.5 for an illustration of how quote styles, aliases, and package options interact. A list of all aliases defined by default is given in table 3.

7.3 Defining package options

The following command creates a new package option based on a key/value syntax. It takes one mandatory argument, the quote style name:

`\DeclareQuoteOption`{⟨*style*⟩}

When using the new option, the name of the quote style will serve as the key. The value may be any style variant defined for the respective style. The package option will select a variant by defining an alias pointing to the desired backend style. This command is available in the configuration file only. See section 7.5 for an illustration of how quote styles, aliases, and package options interact.

7.4 Executing package options

Apart from passing options to this package as it is loaded, you may also execute options using the following command:

`\ExecuteQuoteOptions{`⟨*key=value, ...*⟩`}`

This command permits presetting package options in the configuration file. It may also be used in the document preamble.

7.5 Adding a new quote style

This section will give some comprehensive examples of how to define new quote styles. The examples presented here will only make use of the most basic components a quote style can be composed of. The main point is to illustrate the interaction of quote styles, variants, aliases, and package options. To get started, consider a simple house style which may be selected by way of the package option `style` or the command `\setquotestyle`:

```
\DeclareQuoteStyle{house}
  {\textquotedblleft}{\textquotedblright}
  {\textquoteleft}{\textquoteright}
```

Now suppose that we wanted to add a quote style for an imaginary language called Newspeak and that there were two quote styles commonly used in Newspeak, an official one and an unofficial one. In this case, we need two backend styles implemented as variants of the `newspeak` style, `newspeak/official` and `newspeak/unofficial`:

```
\DeclareQuoteStyle[official]{newspeak}
  {\textquotedblleft}{\textquotedblright}
  {\textquoteleft}{\textquoteright}
\DeclareQuoteStyle[unofficial]{newspeak}
  {\textquotedblright}{\textquotedblleft}
  {\textquoteright}{\textquoteleft}
```

The official variant should be the default for this style. There is no need to copy the definition of the `official` variant to accomplish that. We simply define an alias labeled `newspeak` which points to the desired variant:

```
\DeclareQuoteAlias[official]{newspeak}{newspeak}
```

The reason why we are using variants and aliases instead of two independent styles will become clear in a moment. Suppose that the `babel` package offered support for Newspeak, but this language was known to `babel` as `otherspeak`:

```
\DeclareQuoteAlias{newspeak}{otherspeak}
```

This is an example of a second-level alias pointing to a first-level alias. If the current language is `otherspeak`, the above aliases will be expanded as follows:

`otherspeak` → `newspeak` → `newspeak/official`

We also define a new package option to choose a style variant:

```
\DeclareQuoteOption{newspeak}
```

This will add a new package option with a key called `newspeak`. The value of this option may be any variant of the `newspeak` style defined in the configuration

file. In this example, there are two possible values: `official` and `unofficial`. To select the default or the alternative style for the entire document we use:

```
\usepackage[style=newspeak]{csquotes}
\usepackage[style=newspeak,newspeak=unofficial]{csquotes}
```

To select the default or the alternative style with multilingual support we use:

```
\usepackage[babel]{csquotes}
\usepackage[babel,newspeak=unofficial]{csquotes}
```

The base style must be implemented as an alias in this case since the `newspeak` option will select a variant by redefining and thus overwriting the `newspeak` alias. Since the `otherspeak` alias points to `newspeak` and not directly to any backend style, using the `newspeak` option will also have the desired effect if multilingual support is enabled.

Note that there are two style names which have a special meaning: `default` and `fallback`. The former is an alias pointing to the default quote style used if the multilingual interface is not enabled. The package option `style` and the command `\setquotestyle` will redefine this alias. The latter is a backend style used as a fallback whenever the multilingual interface is enabled but there is no quote style for the current language. It will print bold question marks by default.

7.6 Defining quotes for PDF strings

The following command specifies the quotation marks for PDF strings:

`\DeclarePlainStyle`{⟨*opening outer mark*⟩}{⟨*closing outer mark*⟩}%
{⟨*opening inner mark*⟩}{⟨*closing inner mark*⟩}

This command may be used in the configuration file or in the document preamble. By default, outer quotations get straight double quotes and inner quotations straight single quotes. See section 8.6 for additional hints concerning PDF strings.

7.7 Configuring quotations and citations

The following commands change the default values used by various quotation facilities of this package:

`\SetBlockThreshold`{⟨*integer*⟩}
`\SetBlockEnvironment`{⟨*environment*⟩}
`\SetCiteCommand`{⟨*command*⟩}

`\SetBlockThreshold` changes the number of lines the block quotation facilities use as a threshold when deciding whether a quotation should be set inline or as a block quotation. The default is three lines. `\SetBlockEnvironment` specifies the environment used for block and display quotations. It takes the name of an existing environment as its argument. The default is the `quote` environment provided by most document classes. The argument to `\SetCiteCommand` specifies a replacement for `\cite` which will be used by all integrated quotation facilities to handle citations. It must be a single command which takes one or two optional

Parameter	Command or environment																			
	\enquote	\foreignquote	\hyphenquote	\textquote	\foreigntextquote	\hyphentextquote	\textcquote	\foreigntextcquote	\hyphentextcquote	\blockquote	\foreignblockquote	\hyphenblockquote	\blockcquote	\foreignblockcquote	\hyphenblockcquote	displayquote	foreigndisplayquote	hyphendisplayquote	displaycquote	foreigndisplaycquote	hyphendisplaycquote
Threshold	–	–	–	–	–	–	–	–	–	•	•	•	•	•	•	–	–	–	–	–	–
Environment	–	–	–	–	–	–	–	–	–	•	•	•	•	•	•	•	•	•	•	•	•
Cite command	–	–	–	–	–	–	•	•	•	–	–	–	•	•	•	–	–	–	•	•	•

Table 4: Scope of configurable parameters

arguments followed by a mandatory one, the citation key. The default is `\cite`. The commands affected by these parameters are indicated in table 4.

7.8 Hooks for quotations and citations

Apart from the environment specified with `\SetBlockEnvironment`, which envelops both the quoted text and the citation, the quoted text (excluding the citation) of all long quotations is enclosed in an environment called `quoteblock`. This environment does nothing by default but it may be redefined to format the quoted text. In a similar manner, the text block (excluding the quotation marks) of inline quotations is enclosed in an environment called `quotetext`. Like `quoteblock`, this environment does nothing by default but it may be redefined if additional hooks are required to format the quoted text.

Other aspects of the behavior of the quotation facilities may also be customized at a lower level. All facilities which take a ⟨*cite*⟩ argument will not insert it directly. They pass it to an auxiliary command called `\mkcitation` which may be redefined to format the citation. When doing so, keep in mind that it must take exactly one mandatory argument. `\mkcitation` will only be executed if there is a citation. The default behavior is to separate the citation from the preceding text by an interword space and enclose it in parentheses.

The integrated quotation facilities have slightly different requirements since the `\cite` command may enclose the citation in parentheses or brackets. Therefore, they use `\mkccitation` instead of `\mkcitation`. The default behavior of this command is to separate the citation from the preceding text by an interword space. The default settings are equivalent to the following definitions:

```
\newcommand*{\mkcitation}[1]{␣(#1)}
\newcommand*{\mkccitation}[1]{␣#1}
```

As the block quotation facilities switch between inline and display quotations, changes to the terminal punctuation may be required. The punctuation is controlled by three hooks: `\mkpreblockpunct` is executed before the closing quotation mark. `\mkmidblockpunct` is executed after the closing quotation mark, between the quotation and the citation. `\mkfinblockpunct` is executed after the citation. These hooks serve two purposes. They control the placement of the ⟨*punct*⟩ argu-

Hook	Command or environment																				
	\enquote	\foreignquote	\hyphenquote	\textquote	\foreigntextquote	\hyphentextquote	\textcquote	\foreigntextcquote	\hyphentextcquote	\blockquote	\foreignblockquote	\hyphenblockquote	\blockcquote	\foreignblockcquote	\hyphenblockcquote	displayquote	foreigndisplayquote	hyphendisplayquote	displaycquote	foreigndisplaycquote	hyphendisplaycquote
\mkcitation	–	–	–	●	●	●	–	–	–	●	●	●	–	–	–	●	●	●	–	–	–
\mkccitation	–	–	–	–	–	–	●	●	●	–	–	–	●	●	●	–	–	–	●	●	●
\mkpretextpunct	–	–	–	●	●	●	●	●	●	–	–	–	–	–	–	–	–	–	–	–	–
\mkmidtextpunct	–	–	–	●	●	●	●	●	●	–	–	–	–	–	–	–	–	–	–	–	–
\mkfintextpunct	–	–	–	●	●	●	●	●	●	–	–	–	–	–	–	–	–	–	–	–	–
\mkpreblockpunct	–	–	–	–	–	–	–	–	–	●	●	●	●	●	●	–	–	–	–	–	–
\mkmidblockpunct	–	–	–	–	–	–	–	–	–	●	●	●	●	●	●	–	–	–	–	–	–
\mkfinblockpunct	–	–	–	–	–	–	–	–	–	●	●	●	●	●	●	–	–	–	–	–	–
\mkpredisppunct	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	●	●	●	●	●	●
\mkmiddisppunct	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	●	●	●	●	●	●
\mkfindisppunct	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	●	●	●	●	●	●
quotetext	–	–	–	●	●	●	●	●	●	●	●	●	●	●	●	–	–	–	–	–	–
quoteblock	–	–	–	–	–	–	–	–	–	●	●	●	●	●	●	●	●	●	●	●	●

Table 5: Availability of auxiliary hooks

ment and they may also be used to insert additional punctuation. Like the ⟨*cite*⟩ argument, ⟨*punct*⟩ is not inserted directly. It is passed to all three hooks. When redefining these commands, keep in mind that all of them must take one mandatory argument, but only one of them should insert it. By default, the punctuation is inserted after the citation. This is equivalent to the following definitions:

```
\newcommand*{\mkpreblockpunct}[1]{}
\newcommand*{\mkmidblockpunct}[1]{}
\newcommand*{\mkfinblockpunct}[1]{#1}
```

The text quotation facilities also take a ⟨*punct*⟩ argument whose placement is controlled by `\mkpretextpunct`, `\mkmidtextpunct`, and `\mkfintextpunct`. These commands work like their counterparts for block quotations. Their default behavior is equivalent to the following definitions:

```
\newcommand*{\mkpretextpunct}[1]{}
\newcommand*{\mkmidtextpunct}[1]{}
\newcommand*{\mkfintextpunct}[1]{#1}
```

The hooks `\mkpredisppunct`, `\mkmiddisppunct`, and `\mkfindisppunct` handle the placement of the punctuation argument passed to the display environments. The default behavior is equivalent to the following definitions:

```
\newcommand*{\mkpredisppunct}[1]{}
\newcommand*{\mkmiddisppunct}[1]{}
\newcommand*{\mkfindisppunct}[1]{#1}
```

The difference between `\mkpredisppunct` is `\mkmiddisppunct` is subtle because there are no closing quotation marks in display quotations. The two hooks differ in

that `\mkpredisppunct` is executed before the `quoteblock` environment is closed whereas `\mkmiddisppunct` is executed after this environment. This also applies to `\mkpreblockpunct` and `\mkmidblockpunct` in display mode. Table 5 gives an overview of the facilities affected by a redefinition of the above hooks. See the tutorial for usage examples. Also see section 7.9 for tests which may be useful when redefining the above hooks.

7.9 Additional tests in quotation hooks

The commands in this section increase the flexibility of the hooks discussed in section 7.8. For example, it may be desirable to adjust the format of a citation depending on the way the corresponding quotation is typeset. The following command will test whether the quotation is set inline or as a separate paragraph:

`\ifblockquote`{⟨*true*⟩}{⟨*false*⟩}

This command expands to ⟨*true*⟩ with block and display quotations, and to ⟨*false*⟩ otherwise. It may also be useful to known if the quotation ends with a punctuation mark, especially in the definition of the `\mk...punct` hooks. The following tests provide information about the terminal punctuation:

`\ifquotepunct`{⟨*true*⟩}{⟨*false*⟩}
`\ifquoteterm`{⟨*true*⟩}{⟨*false*⟩}

The first command expands to ⟨*true*⟩ if the quotation ends with any punctuation mark, and to ⟨*false*⟩ otherwise. The second one will only expand to ⟨*true*⟩ if the quotation ends with a terminal punctuation mark (period, exclamation mark, or question mark). The following commands allow for more specific tests:

`\ifquotecolon`{⟨*true*⟩}{⟨*false*⟩}
`\ifquotecomma`{⟨*true*⟩}{⟨*false*⟩}
`\ifquoteexclam`{⟨*true*⟩}{⟨*false*⟩}
`\ifquoteperiod`{⟨*true*⟩}{⟨*false*⟩}
`\ifquotequestion`{⟨*true*⟩}{⟨*false*⟩}
`\ifquotesemicolon`{⟨*true*⟩}{⟨*false*⟩}

Note that all of the above tests are designed for use in the definition of the hooks from section 7.8. They will not yield meaningful results when used anywhere else. There is also a stand-alone test which may be used anywhere:

`\ifstringblank`{⟨*string*⟩}{⟨*true*⟩}{⟨*false*⟩}

This command expands to ⟨*true*⟩ if ⟨*string*⟩ is blank (empty or spaces), and to ⟨*false*⟩ otherwise. This is useful to test for an empty argument in the definition of the `\mk...punct` commands. Note that this test is redundant in the definition of the citation hooks because they are only executed if there is a citation.

Double quotation marks		Single quotation marks	
Command	**Example**	**Command**	**Example**
`\textquotedblleft`	“AaGg“	`\textquoteleft`	‘AaGg‘
`\textquotedblright`	”AaGg”	`\textquoteright`	’AaGg’
`\quotedblbase`	„AaGg„	`\quotesinglbase`	‚AaGg‚
`\guillemotleft`	«AaGg«	`\guilsinglleft`	‹AaGg‹
`\guillemotright`	»AaGg»	`\guilsinglright`	›AaGg›

Table 6: Quotation marks included in T1 and LY1 encoding

8 Hints and caveats

8.1 Input encodings

The active quotes provided by this package may depend on or benefit from the `inputenc` package under certain circumstances. As long as the active quotes are in the range 0–127, there is no benefit in loading `inputenc`. If you are using an 8-bit input encoding such as `latin1`, `inputenc` is required for the quotes to function properly in a verbatim context. It should therefore be loaded before any active quotes are allocated (not necessarily before this package is loaded). The UTF-8 support of this package builds on the `utf8` module of the inputenc package. When using this encoding, ensure that inputenc is loaded with the `utf8` option. Do not use the `utf8x` option as this would implicitly load the `ucs` package which is not supported by `csquotes`. UTF-8 encoding will be detected automatically. All commands discussed in section 4 work as usual with this encoding. See also section 8.5.

8.2 Output encodings

The OT1 font encoding, the default output encoding of LaTeX, merely includes the quotation marks used in English. You will need to switch to T1 or LY1 encoding in order to get guillemets or baseline quotation marks. This package deliberately refrains from providing any workarounds for the OT1 legacy encoding. If you need T1 encoding for some of the quotation marks, you will most likely need it anyway to get proper hyphenation for the respective language. See table 6 for a list of common quotation marks included in both T1 and LY1 encoding.

8.3 Valid active quotes

In general, an active quote may be any single character with category code 12 or 13, or any multibyte UTF-8 sequence representing a single character. However, there are a few exceptions: numbers, punctuation marks, the apostrophe, the hyphen, and all characters which are part of the LaTeX syntax are rejected. In sum, the following characters will be considered as reserved by this package: `A-Z a-z 0-9 . , ; : ! ? ' - # $ % & ^ _ ` ~ \ @ * { } [ ]`

8.4 Invalid nesting and unbalanced active quotes

Every quotation forms a group which includes both the quoted piece of text and the quotation marks. This package tracks the nesting level of all quotations and thus allows for basic validation. If quotations are nested in an invalid way, it will issue an error message. Keep in mind that the active quotes are more than a conve-

nient way to enter quotation marks. They are fully-fledged markup elements which imply grouping as well, hence they must always be balanced and must not interfere with other group boundaries. This package ensures that an error is triggered if quotes are unbalanced or nested in an invalid way. However, note that packages generally can not catch low-level errors caused by grouping mistakes, nor do they have any control over the wording of generic error messages.

8.5 Active quotes in special contexts

The commands provided by this package are designed for use in text mode. If you inadvertently use them in math mode, they will issue an error message. Note that all active quotes retain their original function in math mode. It is perfectly possible to use a character like the greater-than symbol as an active quote without interfering with math mode.

In a verbatim context, the active quotes will normally be disabled. If a character is in the range 128–255, however, its original function is restored so that the `inputenc` package may handle it in verbatim environments. This implies that `inputenc` must be loaded before any active quotes are allocated. This feature is available with the standard verbatim environments as well as those provided by (or defined via) the packages `verbatim`, `fancyvrb`, `moreverb`, and `alltt`. This also applies to the `\verb` command and the `shortvrb` package. The `listings` package provides dedicated support for extended input encodings. When using this package, activate its 'extended characters' option and specify the input encoding. As of this writing, the `listings` package does not support UTF-8 encoding.

Some care is still required when choosing active quotes. Note that you normally cannot use active characters in the argument to commands expecting a string of characters, such as `\input`, `\label`, or `\cite`. There are two packages which try to remedy this situation: the `babel` package and the `underscore` package (when loaded with the `strings` option). Both packages redefine several standard commands affected by this general problem. If any one of these packages is loaded, `csquotes` will take advantage of all improvements automatically. Unfortunately, both packages patch a different set of commands and neither one covers all possibly vulnerable commands.

8.6 PDF strings and `hyperref` support

This package interfaces with the `hyperref` package as PDF strings such as bookmarks are generated. See section 7.6 on how to configure the quotation marks used in PDF strings. Support for PDF strings is only available with the basic facilities presented in sections 3.1 and 3.2 as well as 4.1 and 4.2. Be advised that the way `hyperref` builds PDF strings imposes severe limitations on the capabilities of all commands. Most notably, the nesting level of quotations cannot be tracked in this context. Nested quotations will generally get outer marks, but you may use starred commands or active inner quotes to request inner marks explicitly. If quotation marks are to be included in the document properties of a PDF file, you must use `\hypersetup` to specify the strings. The replacement mechanism will not function within the optional argument to `\usepackage`. For information about PDF strings see the `hyperref` documentation.

8.7 Footnotes within quotations

This package will automatically reset the nesting level within any footnote included in a quotation. If the babel package has been loaded, it will also reset the language. The language of the footnote text including the hyphenation patterns will match the language of the text surrounding the quotation. This applies to parboxes, minipages, and floats as well.

8.8 Using csquotes with babel's shorthands

The commands discussed in section 9.4 may be combined with the shorthands of the babel package such that babel provides the user interface and csquotes the backend. For example, the German module of the babel package defines, amongst other things, the shorthands "‘ and "’. Such shorthands are input aids, i. e., physical markup elements with a fixed definition. Typing "‘ is a short way of saying \quotedblbase but it is not different in concept. These shorthands can be transformed into 'smart quotes' which behave like \enquote. Here is a simple ad hoc solution suitable for documents with only one language:

```
\documentclass{article}
\usepackage[german]{babel}
\usepackage[babel=once]{csquotes}
\defineshorthand{"`}{\openautoquote}
\defineshorthand{"'}{\closeautoquote}
\begin{document}
...
\end{document}
```

It is possible to move such definitions to csquotes.cfg. In this case, the code is slightly more complex because it needs to be more general:

```
\AtEndPreamble{%
  \@ifpackageloaded{babel}
    {\iflanguage{german}
       {\declare@shorthand{german}{"`}{\openautoquote}%
        \declare@shorthand{german}{"'}{\closeautoquote}}
       {}}
    {}}
```

The above code redefines the shorthands only if the babel package has been loaded and the main language of the document is german. Note that babel's shorthands are language-specific. The way they are configured and handled is technically and conceptually different from the active quotes discussed in section 4. Active quotes are defined globally and automatically adapt to the current language. With babel, each language has its own set of shorthands. Also note that babel uses \AtBeginDocument to initialize the main document language, including the corresponding shorthands. We use \AtEndPreamble to defer the code until the end of the preamble. This way, we can be sure that babel has been loaded but that the main document language has not been initialized yet. See the babel manual

for further details. The `\AtEndPreamble` command is provided by the `etoolbox` package.

8.9 Miscellaneous notes about the predefined styles

All variants of the `french` style use spaced out guillemets as outer marks. The style variant `quotes` uses double quotes as inner marks. The starred variant `quotes*` is similar to its regular counterpart except that it will also space out the inner marks. The `guillemets` variant employs spaced out guillemets on all levels. It will also insert guillemets at the beginning of every paragraph inside a quotation spanning multiple paragraphs. In addition to that, two adjoining marks at the end of a quotation are replaced by a single one; if two nested quotations end simultaneously, the second closing mark is omitted automatically. The starred variant `guillemets*` is similar to its regular counterpart, differing only in the middle mark inserted at the beginning of every paragraph. The regular variant uses a left-pointing guillemet whereas the starred one uses a right-pointing one.

9 Author interface

The following sections discuss the programmer interface to the `csquotes` package as well as some details of the implementation. They are intended for class and package authors who want to interface with this package.

9.1 Controlling active quotes

The author commands in this section behave essentially like the corresponding user commands discussed in section 4.5. The only difference is that they work quietly behind the scenes without writing any notices to the transcript file. The scope of these commands is local so that all changes may be confined to a group. Note that the active quotes are enabled at the beginning of the document body. Under no circumstances will this package make any characters active in the document preamble. You will only need the following commands when dealing with active quotes at the beginning of or in the document body.

`\@enablequotes` This command enables all characters allocated as active quotes. It also restores their definitions if they were disabled or accidentally overwritten. With single-byte encodings, this command (re)defines all allocated characters and makes them active. With UTF-8 encoding, it redefines the internal macro used by the `inputenc` package to typeset the respective UTF-8 sequence (`\u8:`⟨*character*⟩). UTF-8 characters in the range 0–127 are handled as with single-byte encodings.

`\@disablequotes` This command restores the *status quo ante* of all active quotes. With single-byte encodings, there are two possible cases. (1) If a character had already been active when it was allocated as active quote, its former definition is restored. (2) If a character had not been active when it was allocated, its former category code is restored. With UTF-8 encoding, this command restores the former definition of the internal macro used by the `inputenc` package to typeset the respective UTF-8 sequence. UTF-8 characters in the range 0–127 are handled as with single-byte encodings.

\@verbatimquotes For verbatim environments and similar applications, use this command rather than \@disablequotes. It redefines the active quotes in a way that is better suited for verbatim typesetting. With single-byte encodings, it will do one of the following things. (1) If a character is in the range 0–127, it is redefined such that it expands to itself with category code 12. (2) If a character is in the range 128–255, there are two possibilities. (a) If it had already been active when it was allocated, its former definition is restored. (b) If it had not been active before, it is redefined such that it expands to itself with its former category code.
Characters in the range 0–127 are added to the \dospecials list. Characters in the range 128–255 remain active, permitting the inputenc package to typeset them verbatim (due to case 2a, which implies that you must load inputenc before allocating active quotes). Case 2b is usually undesirable in verbatim environments. If inputenc is loaded, however, this should not happen. With UTF-8 encoding, this command restores the former definition of the internal macro used by the inputenc package to typeset the respective UTF-8 sequence. UTF-8 characters in the range 0–127 are handled as with single-byte encodings.
Due to case 1, \@verbatimquotes itself is independent of any \dospecials processing. You may typeset all active quotes verbatim by using this command exclusively. The advantage of this approach is that it does not require any category code changes, hence this command may also be used to modify an argument after it has been read. Also note that the standard LaTeX verbatim environments as well as all environments provided by or defined via the packages verbatim, fancyvrb, moreverb, and alltt are catered for automatically. This also applies to the \verb command and the shortvrb package.

\@deletequotes This command implicitly executes \@disablequotes and deallocates all active quotes, which results in a complete reset of all active quotes so that they may be newly defined. This command should be used with care because the reset is not visible to the user. Using \DeleteQuotes may be preferable.

9.2 Active quotes in a strings-only context

A possible problem with active characters are strings-only contexts, i. e. cases in which an active character is used in the formation of a control sequence name or as a plain string. A typical example is a command like \label which expects a string of characters. Any active character may break \label when used in its argument. There are two packages which try to remedy this situation, albeit in different ways: babel and underscore.

The babel package defines the switch \if@safe@actives and patches several standard commands such that the switch is set to true while they process their arguments. The approach taken by the underscore package is slightly different. If underscore is loaded with the strings option, it patches several commands such that \protect is equivalent to \string while the arguments are processed. If any one of these packages is loaded, csquotes will take advantage of that automatically. Unfortunately, both packages patch a different set of commands and neither one covers all possibly vulnerable commands. If babel is loaded, for example, you may use active quotes in the argument of \label, but not in the argument

of `\input`. If you load `underscore` with its `strings` option, active quotes may also be used in the argument of `\input`.

When writing a package which may have to process user-supplied arguments in a strings-only context, there are two ways to deal with active quotes. Taking the approach of the `babel` package, you may do the following:

```
\let\if@safe@actives\iftrue
```

This is best done in a group. If grouping is not feasible, you must ensure that the switch is properly restored. In contrast to using `\@safe@activestrue`, this approach works even if `babel` is not loaded. However, note that you must take three states into account when restoring the switch in this case: true, false, and undefined. Taking the approach of the `underscore` package, you may also do the following:

```
\let\@@protect\protect
\let\protect\string
```

This could either be done in a group or without any grouping, but followed by `\restore@protect`. The first approach works with the active characters of the `babel` and the `underscore` packages. The second one works with the `underscore` and the `at` packages. Unfortunately, the active characters of the `inputenc` package support neither of the above-mentioned techniques. As far as `csquotes` is concerned, it does not matter which approach you take. In both cases all active quotes expand to themselves with category code 12. UTF-8 encoded active quotes expand to a string of characters with category code 12. This string will be valid UTF-8. In a verbatim `\write` operation, you should employ one of the techniques discussed in this section rather than `\@verbatimquotes`, which is geared to verbatim typesetting.

9.3 Block quotations

The block quotation facilities need to typeset all quotations twice. The first pass is required to measure the length of the quotation. The actual typesetting takes place on the second pass, in a format depending on the result of the first one. In order to prevent any side-effects of the first (trial) pass, the `csquotes` package (1) performs the first pass inside a group, (2) employs checkpointing to freeze all LaTeX counters, and (3) sets `\if@filesw` to `false`. However, it can not prevent side-effects caused by commands that (1) make any global assignments which are not overwritten on the second pass (for example, by way of `\g@addto@macro`), (2) increment counters globally in a way that circumvents LaTeX's counter commands, or (3) do not check `\if@filesw` every time they are about to write to an auxiliary file. If you observe any malfunctions related to the trial pass (for example, if counters are incremented twice or if an item appears twice in a list), use `\BlockquoteDisable` to redefine or disable the affected command temporarily.

`\BlockquoteDisable{`⟨*code*⟩`}`

The ⟨*code*⟩ may be arbitrary LaTeX code which redefines vulnerable commands locally such that they work differently during the trial pass. The ⟨*code*⟩ itself should

obviously not include any global assignments. This solution should be considered as a last ressort but may be the quickest way to fix a vulnerable package. Note that there is no need to escape parameter characters by doubling them in the ⟨*code*⟩ argument. Simply use this command like `\AtBeginDocument` and similar hooks.

9.4 Automatic quotation marks

The commands in this section provide access to the automatic quotation facilities at a slightly lower level than the user commands in sections 3.1 and 4.1. In contrast to the commands discussed in section 9.5, the facilities in this section are fully-fledged markup elements which verify the nesting level and issue an error if quotations are nested in an invalid way. They form groups and must always be balanced, see section 8.4 for details. In other words, the facilities in this section are semantic markup elements, the ones in section 9.5 are physical markup elements.

`\openautoquote` Opens a nestable quotation.
`\closeautoquote` Closes a nestable quotation.

In terms of their function, the above commands correspond to the regular versions of `\enquote` and `\MakeAutoQuote`. The following commands correspond to the starred variants `\enquote*` and `\MakeAutoQuote*`:

`\openinnerquote` Opens an inner quotation.
`\closeinnerquote` Closes an inner quotation.

The above commands may be used to implement an alternative user interface. You can combine them with the shorthands of the `babel` package such that `babel` provides the user interface and `csquotes` the backend. See section 8.8 and the `babel` manual for details.

9.5 Internal quotation marks

The commands in this section print the quotation marks of the current style, as defined with `\DeclareQuoteStyle`, without any grouping or nesting control. The quotation marks reflect all changes to the quotation style. If the multilingual interface is enabled, they are also synced with the current language.

`\textooquote` Prints the opening outer quotation mark of the currently active quote style.
`\textcoquote` Prints the closing outer quotation mark.
`\textmoquote` Prints the middle outer quotation mark.

`\textoiquote` Prints the opening inner quotation mark.
`\textciquote` Prints the closing inner quotation mark.
`\textmiquote` Prints the middle inner quotation mark.

Note that the initialization hooks for the respective quotation style are not executed automatically. They may be accessed separately:

`\initoquote` Executes the outer initialization hook.
`\initiquote` Executes the inner initialization hook.

The scope of these hooks should always be confined to a group.

10 Revision history

This revision history is a list of changes relevant to users of this package. Changes of a more technical nature which do not affect the user interface or the behavior of the package are not included in the list. The numbers on the right indicate the relevant section of this manual.

4.1 2008-04-11

Fixed timing issue with active quotes introduced in 4.0

4.0 2008-03-02

e-TeX now mandatory requirement 1.2
New dependency on `etoolbox` package 1.2
Added package option `spanish` 2
Added variant `mexican` to style `spanish` 2
Removed variant `oldstyle` from `english` style 2
Removed variant `oldstyle` from `french` style 2
Removed variant `imprimerie` from `french` style 2
Expanded documentation 8.8
Added `\openautoquote` and `\closeautoquote` 9.4
Added `\openinnerquote` and `\closeinnerquote` 9.4
Moved predefined styles, variants, options to `csquotes.def`
Added more hints and examples to `csquotes.cfg`
Added extended PDF bookmarks to this manual

3.8 2008-01-05

Added variant `guillemets*` to style `swedish` 2
Added language alias `australian` 7.2
Added language alias `newzealand` 7.2
Internal improvements

3.7 2007-03-25

Added package option `babel=try` 2.2
Added package option `babel=tryonce` 2.2
Added `\MakeAutoQuote*` 4.1
Added `\MakeForeignQuote*` 4.2
Added `\MakeHyphenQuote*` 4.2
Added `\mkpretextpunct` 7.8
Added `\mkpreblockpunct` 7.8
Added `\mkpredisppunct` 7.8
Dropped compatibility code for `\blockcite` legacy command
Internal updates for `biblatex` package

3.6 2006-11-09

Added `\BlockquoteDisable` 9.3
Fix for `amsmath` package (active quotes in `split` and other environments)

Fix for `endnotes` package (endnotes in block quotations)
Revised Spanish quote style

3.5 2006-08-24

Exchanged definitions of French `quotes` and `quotes*` variants 8.9
Internal updates for `inputenc` 1.1 b (2006-05-05)

3.4 2006-04-02

Stricter validation of user-defined active characters 8.3
Author interface now documented in this manual 9
Added documentation of `\@enablequotes` 9.1
Added documentation of `\@disablequotes` 9.1
Added documentation of `\@verbatimquotes` 9.1
Added documentation of `\@deletequotes` 9.1
Added documentation concerning string handling 9.2
Added documentation of interface to internal marks 9.5

3.3 2006-02-27

Added support for UTF-8 encoded active quotes 8.1
Modified active quotes, category codes 7, 8 no longer valid 8.3
Modified delimiters, category codes 3, 4, 7, 8 no longer valid 8.3
Active quotes may now be defined in the document body 4.5
Renamed `\RestoreQuotes` to `\EnableQuotes` 4.5
Added `\DeleteQuotes` 4.5
Added `\VerbatimQuotes` 4.5
Added `\ExecuteQuoteOptions` 7.4
Added package option `babel=once` 2.2
Added new style variant for French 8.9
Improved nesting control of active block quotes
Made active block quotes robust

3.2 2005-12-05

Added quote style for Spanish
Fixed bug in `hyperref` interface

3.1 2005-08-29

Added `\textquote` 3.3
Added `\foreigntextquote` 3.4
Added `\hyphentextquote` 3.4
Renamed `\cquote` to `\textcquote` 5.1
Renamed `\foreigncquote` to `\foreigntextcquote` 5.2
Renamed `\hyphencquote` to `\hyphentextcquote` 5.2
Extended `\textcquote` 5.1
Extended `\foreigntextcquote` 5.2
Extended `\hyphentextcquote` 5.2
Modified environment `displayquote` 6.1

Modified environment foreigndisplayquote 6.1
Modified environment hyphendisplayquote 6.1
Extended environment displaycquote 6.2
Extended environment foreigndisplaycquote 6.2
Extended environment hyphendisplaycquote 6.2
Added \mkmidtextpunct 7.8
Added \mkfintextpunct 7.8
Added \mkmiddisppunct 7.8
Added \mkfindisppunct 7.8
Added auxiliary environment quotetext 7.8
Added detection of paragraphs to all block quotation facilities 3.5
\ifquote... now usable in \mkcitation and \mkccitation 7.9
Terminal punctuation now evaluated by all quotation facilities
Prevent undesirable ?‘ and !‘ ligatures in T1 encoding
Always adjust space factor codes of backend quotes

3.0 2005-07-14

Extended \blockquote 3.5
Extended \foreignblockquote 3.6
Extended \hyphenblockquote 3.6
Extended \setquotestyle 3.7
Added \cquote 5.1
Added \foreigncquote 5.2
Added \hyphencquote 5.2
Added \blockcquote 5.3
Added \foreignblockcquote 5.4
Added \hyphenblockcquote 5.4
Added environment displayquote 6.1
Added environment foreigndisplayquote 6.1
Added environment hyphendisplayquote 6.1
Added environment displaycquote 6.2
Added environment foreigndisplaycquote 6.2
Added environment hyphendisplaycquote 6.2
Modified \DeclarePlainStyle 7.6
Added \SetCiteCommand 7.7
Renamed \blockcite to \mkcitation 7.8
Added \mkccitation 7.8
Added \mkmidblockpunct 7.8
Added \mkfinblockpunct 7.8
Added \ifquotepunct 7.9
Added \ifquoteterm 7.9
Added \ifquoteperiod 7.9
Added \ifquotecomma 7.9
Added \ifquotesemicolon 7.9
Added \ifquotecolon 7.9
Added \ifquoteexclam 7.9

Added `\ifquotequestion` 7.9
Added `\ifstringblank` 7.9
Added evaluation of terminal punctuation within block quotations
With `\nonfrenchspacing`, adjust space factor codes of backend quotes
Improved nesting control when running under e-TeX

2.8 2005-05-11

Added `\DisableQuotes` 4.5
Fixed bug causing kerning restoration to fail in some rare cases

2.7 2005-04-13

Use the font's kerning pairs for adjoining quotes, if available 7.1
Renamed `\setblockthreshold` to `\SetBlockThreshold` 7.7
Renamed `\setblockenvironment` to `\SetBlockEnvironment` 7.7
Provided more useful default definition of `\blockcite` 7.8
Improved handling of adjoining quotes with respect to line breaking
When restoring active quotes, restore catcodes of delimiters as well
Improved workaround for `\uppercase` and some `babel` languages
Issue error message on quote mismatch regardless of `strict` option
Issue `hyperref` warning with block quotation commands in PDF strings
Fixed bug in `\DeclareQuoteStyle` and `\DeclareQuoteAlias`

2.6 2005-02-24

Always reset quote style, even for inner quotations
Fixed bug preventing hyphenation in certain places

2.5 2004-12-04

Added `\MakeBlockQuote` 4.3
Added `\MakeForeignBlockQuote` 4.4
Added `\MakeHyphenBlockQuote` 4.4
Added `\ifblockquote` 7.9
Modified `\blockquote` 3.5
Modified `\foreignblockquote` 3.6
Modified `\hyphenblockquote` 3.6
Changed default threshold for block quotations 7.7
Improved math mode compatibility 8.5
Improved verbatim compatibility 8.5
Improved backend and active character handling
Improved validation of user-defined active characters
Fixed bug suppressing kerning after block quotations
Issue error message with nested block quotations

2.4 2004-11-01

Prevent use of `\RestoreQuotes` in preamble 4.5
Fixed bug causing premature expansion of backend quote macros

Fixed bug suppressing kerning before closing quotes

2.3 2004-09-18

Reduced default kerning between adjoining curved quotes
Fixed bug with `\DeclareQuoteStyle` in preamble

2.2 2004-07-13

Extended `\DeclareQuoteStyle` 7.1
Added initialization hook for inner quotations 7.1
Added support for middle inner quotes 7.1
Rearranged French quote styles, removing two variants 8.9
Added new style variant for French 8.9
Fixed bug causing stacking of reset hook for footnotes
Fixed bug preventing hyphenation in certain places
Fixed kerning issue specific to EC fonts

2.1 2004-06-15

Added auxiliary environment `quoteblock` 7.8
Added support for language reset in footnotes 8.7
Disable active characters in `\verb` and `verbatim` 8.5
Disable active characters in `\index` and `\glossary` 8.5
Added package option and style variants for Norwegian
Removed some uncertain quote styles and aliases
Rearranged quote styles and aliases

2.0 2004-06-04

Added `\blockquote` 3.5
Added `\foreignblockquote` 3.6
Added `\hyphenblockquote` 3.6
Added `\setblockthreshold` 7.7
Added `\setblockenvironment` 7.7
Added auxiliary command `\blockcite` 7.8
Extended `\DeclareQuoteStyle` 7.1
Added initialization hook for outer quotations 7.1
Added support for middle outer quotes 7.1
Added support for kerning between adjoining quotes 7.1
Disable active characters in math and display math mode 8.5
Revised and improved error recovery 2.1
Added package option `strict` 2.1
Added package option and new style variants for French
Added package option and new style variant for Italian
Added new style variant for English

1.7 2004-05-14

Added `\setquotestyle` 3.7
Modified `\DeclarePlainStyle` 7.6

Improved quote handling in PDF strings 8.6
Amended default French quote style 8.9

1.5 2004-02-27

Reset quote nesting level in footnotes within quotations 8.7

1.4 2003-12-13

Added `\MakeForeignQuote` 4.2
Added `\MakeHyphenQuote` 4.2
Added `\RestoreQuotes` 4.5
Improved `hyperref` interface 8.6

1.0 2003-09-14

Initial public release