

The kvsetkeys package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2006/10/19 v1.1

Abstract

Package kvsetkeys provides `\kvsetkeys`, a variant of package keyval's `\setkeys`. It allows to specify a handler that deals with unknown options. Active commas and equal signs may be used (e.g. see `babel`'s shorthands) and only one level of curly braces is removed from the values.

Contents

1	Documentation	1
1.1	Normalizing key value lists	2
1.2	Parsing key value lists	2
1.3	Processing key value pairs	3
1.4	Default family handler	3
1.5	Do it all	3
2	Example	3
3	Implementation	4
3.1	Identification	4
3.2	Normalizing key value lists	6
3.3	Parsing key value lists	8
3.4	Processing key value pairs	9
3.5	Error handling	10
3.6	Do it all	10
4	Installation	11
4.1	Download	11
4.2	Bundle installation	11
4.3	Package installation	11
4.4	Refresh file name databases	11
4.5	Some details for the interested	11
5	References	12
6	History	12
	[2006/03/06 v1.0]	12
	[2006/10/19 v1.1]	12
7	Index	12

1 Documentation

`\kvsetkeys` can be used as replacement for `keyval`'s `\setkeys`. Also it uses the same syntax. Before I describe `\kvsetkeys` in more detail, first I want to explain, how this package deals with key value lists. For the package also provides low level interfaces that can be used by package authors.

1.1 Normalizing key value lists

`\kv@normalize {⟨key value list⟩}`

Specifying key value lists, the user usually wants to have nice formatted source code, e.g.:

```
\hypersetup{
  pdftitle    = {...},
  pdfsubject  = {...},
  pdfauthor   = {...},
  pdfkeywords = {...},
  ...
}
```

Thus there can be spaces around keys, around = or around the value. Also empty entries are possible by too many commas. Therefore these spaces and empty entries are silently removed by package `keyval` and this package. Whereas the contents of the value can be protected by curly braces, especially if spaces or commas are used inside, a key name must not use spaces or other syntax characters.

`\kv@normalize` takes a key value list and performs the cleanup:

- Spaces are removed.
- Syntax characters (comma and equal sign) that are active are replaced by the same characters with standard catcode. (Example: `babel`'s language option `turkish` uses the equal sign as active shorthand character.)

The result is stored in `\kv@list`, e.g.:

```
\kv@list → ,pdftitle={},pdfsubject={},...,
```

Empty entries are removed later. In fact it adds a comma at the begin and end to protect the last value and an easier implementation.

1.2 Parsing key value lists

`\kv@parse {⟨key value list⟩} {⟨processor⟩}`

It is easier to parse a normalized list, thus `\kv@parse` normalizes the list and calls `\kv@parse@normalized`.

`\kv@parse@normalized {⟨key value list⟩} {⟨processor⟩}`

Now the key value list is split into single key value pairs. For further processing the key and value are given as arguments for the `⟨processor⟩`:

```
⟨processor⟩ {⟨key⟩} {⟨value⟩}
```

Also key and value are stored in macro names:

- `\kv@key` stores the key.
- `\kv@value` stores the value or if the value was not specified it has the meaning `\relax`.

The behaviour in pseudo code:

```
foreach (⟨key⟩, ⟨value⟩) in (⟨key value list⟩)
  \kv@key := ⟨key⟩
  \kv@value := ⟨value⟩
  ⟨processor⟩ {⟨key⟩} {⟨value⟩}
```

1.3 Processing key value pairs

`\kv@processor@default {⟨family⟩} {⟨key⟩} {⟨value⟩}`

There are many possibilities to process key value pairs. `\kv@processor@default` is the processor used in `\kvsetkeys`. It reimplements and extends the behaviour of `keyval`'s `\setkeys`. In case of unknown keys `\setkeys` raise an error. This processor, however, calls a handler instead, if it is provided by the family.

The behaviour in pseudo code:

```
if ⟨key⟩ exists
  call the keyval code of ⟨key⟩
else
  if ⟨handler⟩ for ⟨family⟩ exists
    ⟨handler⟩ {⟨key⟩} {⟨value⟩}
  else
    raise unknown key error
fi
fi
```

1.4 Default family handler

`\kv@processor@default` calls `⟨handler⟩`, the default handler for the family, if the key does not exist in the family. The handler is called with two arguments, the key and the value. It can be defined with `\kv@set@family@handler`:

`\kv@set@family@handler {⟨family⟩} {⟨handler definition⟩}`

This sets the default family handler for the keyval family `⟨family⟩`. Inside `⟨handler definition⟩` #1 stands for the key and #2 is the value. Also `\kv@key` and `\kv@value` can be used for the key and the value. If the value is not given, `\kv@value` has the meaning `\relax`.

1.5 Do it all

`\kvsetkeys {⟨family⟩} {⟨key value list⟩}`

The work is done by the previous commands. `\kvsetkeys` just calls them:

```
\kv@parse {⟨key value list⟩} {\kv@processor@default {⟨family⟩}}
```

Thus you can replace `\setkeys` of package `keyval` by the key value parser of this package:

```
\renewcommand*{\setkeys}{\kvsetkeys}
or
\let\setkeys\kvsetkeys
```

2 Example

The following example prints a short piece of HTML code using the tabbing environment for indenting purpose and a key value syntax for specifying the attributes of an HTML tag. The example illustrates the use of a default family handler.

```
1 ⟨*example⟩
2 \documentclass{article}
3 \usepackage[T1]{fontenc}
4 \usepackage{kvsetkeys}
```

```

5 \usepackage{keyval}
6
7 \makeatletter
8 \newcommand*{\tag}[2] [] {%
9   % #1: attributes
10  % #2: tag name
11  \begingroup
12    \toks@={}%
13    \let\@endslash\@empty
14    \kvsetkeys{tag}{#1}%
15    \texttt{%
16      \textless #2\the\toks@\@endslash\textgreater
17    }%
18  \endgroup
19 }
20 \kv@set@family@handler{tag}{%
21   % #1: key
22   % #2: value
23   \toks@\expandafter{%
24     \the\toks@
25     \space
26     #1=\string"#2\string"%
27   }%
28 }
29 \define@key{tag}{/} [] {%
30   \def\@endslash{/}%
31 }
32 \makeatother
33
34 \begin{document}
35 \begin{tabbing}
36   \mbox{}\qqquad=\qqquad=\kill
37   \tag{html}\|
38   \>\dots\|
39   \>\tag[border=1]{table}\|
40   \>\>\tag[width=200, span=3, /]{colgroup}\|
41   \>\>\dots\|
42   \>\tag{/table}\|
43   \>\dots\|
44   \tag{/html}\|
45 \end{tabbing}
46 \end{document}
47 \end{example}

```

3 Implementation

3.1 Identification

```

48 (*package)

```

Reload check, especially if the package is not used with L^AT_EX.

```

49 \begingroup
50   \catcode44 12 % ,
51   \catcode45 12 % -
52   \catcode46 12 % .
53   \catcode58 12 % :
54   \catcode64 11 % @
55   \expandafter\let\expandafter\x\csname ver@kvsetkeys.sty\endcsname
56   \ifcase 0%
57     \ifx\x\relax % plain
58     \else
59       \ifx\x\empty % LaTeX

```

```

60     \else
61     1%
62     \fi
63     \fi
64 \else
65 \expandafter\ifx\csname PackageInfo\endcsname\relax
66 \def\x#1#2{%
67 \immediate\write-1{Package #1 Info: #2.}%
68 }%
69 \else
70 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
71 \fi
72 \x{kvsetkeys}{The package is already loaded}%
73 \endgroup
74 \expandafter\endinput
75 \fi
76 \endgroup
Package identification:
77 \begingroup
78 \catcode40 12 % (
79 \catcode41 12 % )
80 \catcode44 12 % ,
81 \catcode45 12 % -
82 \catcode46 12 % .
83 \catcode47 12 % /
84 \catcode58 12 % :
85 \catcode64 11 % @
86 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
87 \def\x#1#2#3[#4]{\endgroup
88 \immediate\write-1{Package: #3 #4}%
89 \xdef#1{#4}%
90 }%
91 \else
92 \def\x#1#2[#3]{\endgroup
93 #2[#{#3}]%
94 \ifx#1\relax
95 \xdef#1{#3}%
96 \fi
97 }%
98 \fi
99 \expandafter\x\csname ver@kvsetkeys.sty\endcsname
100 \ProvidesPackage{kvsetkeys}%
101 [2006/10/19 v1.1 Key value parser with default handler support (H0)]
102 \expandafter\edef\csname KVS@endinput\endcsname{%
103 \catcode39 \the\catcode39 % '
104 \catcode44 \the\catcode44 % ,
105 \catcode61 \the\catcode61 % =
106 \catcode64 \the\catcode64 % @
107 \catcode94 \the\catcode94 % ^
108 \catcode96 \the\catcode96 % '
109 \catcode126 \the\catcode126 % ~
110 \relax
111 \noexpand\endinput
112 }
113 \catcode39 12 % '
114 \catcode44 12 % ,
115 \catcode61 12 % =
116 \catcode64 11 % @
117 \catcode94 7 % ^
118 \catcode96 12 % '
119 \catcode126 13 % ~
120 \def\KVS@empty{}

```

```
121 \long\def\@ReturnAfterFi#1\fi{\fi#1}
```

3.2 Normalizing key value lists

`\kv@normalize`

```
122 \def\kv@normalize#1{%
123   \begingroup
124     \toks@{\,#1}%
125     \KVS@comma
126     \KVS@equal
127     \KVS@spaceA
128     \KVS@spaceB{ }%
129     \KVS@spaceC
130     \KVS@spaceD{ }%
131     \xdef\kv@global{\the\toks@}%
132   \endgroup
133   \let\kv@list\kv@global
134 }
```

`\KVS@comma` Converts active commas into comma with catcode other. Also adds a comma at the end to protect the last value for next cleanup steps.

```
135 \begingroup
136   \lccode'\,='\'%
137   \lccode'\~='\'%
138 \lowercase{\endgroup
139   \def\KVS@comma{%
140     \toks@\expandafter{\expandafter}\expandafter
141     \KVS@@comma\the\toks@\KVS@nil
142   }%
143   \def\KVS@@comma#1~#2\KVS@nil{%
144     \toks@\expandafter{\the\toks@#1,}%
145     \toks2{#2}%
146     \edef\x{\the\toks2}%
147     \ifx\x\KVS@empty
148       \else
149         \@ReturnAfterFi{%
150           \KVS@@comma#2\KVS@nil
151         }%
152       \fi
153   }%
154 }
```

`\KVS@equal` Converts active equal signs into catcode other characters.

```
155 \begingroup
156   \lccode'\,='\'%
157   \lccode'\~='\'%
158 \lowercase{\endgroup
159   \def\KVS@equal{%
160     \toks@\expandafter{\expandafter}\expandafter
161     \KVS@@equal\the\toks@\KVS@nil
162   }%
163   \def\KVS@@equal#1~#2\KVS@nil{%
164     \edef\x{\the\toks@}%
165     \ifx\x\KVS@empty
166       \toks@{#1}%
167     \else
168       \toks@\expandafter{\the\toks@=#1}%
169     \fi
170     \toks2{#2}%
171     \edef\x{\the\toks2}%
172     \ifx\x\KVS@empty
173     \else
```

```

174     \@ReturnAfterFi{%
175     \KVS@@equal#2\KVS@nil
176     }%
177     \fi
178 }%
179 }

```

\KVS@spaceA Removes one space after the equal sign. In theory also several spaces could be removed, but this is not really necessary, because T_EX usually collapses several spaces to one already.

```

180 \def\KVS@spaceA{%
181   \toks@\expandafter{\expandafter}\expandafter
182   \KVS@@spaceA\the\toks@= \KVS@nil
183 }
184 \def\KVS@@spaceA#1= #2\KVS@nil{%
185   \edef\x{\the\toks@}%
186   \ifx\x\KVS@empty
187     \toks@{#1}%
188   \else
189     \toks@\expandafter{\the\toks@=#1}%
190   \fi
191   \toks2{#2}%
192   \edef\x{\the\toks2}%
193   \ifx\x\KVS@empty
194     \else
195       \@ReturnAfterFi{%
196       \KVS@@spaceA#2\KVS@nil
197       }%
198   \fi
199 }

```

\KVS@spaceB Removes one space before the comma.

```

200 \def\KVS@spaceB#1{%
201   \toks@\expandafter{\expandafter}\expandafter
202   \KVS@@spaceB\the\toks@#1,\KVS@nil
203 }
204 \def\KVS@@spaceB#1 ,#2\KVS@nil{%
205   \edef\x{\the\toks@}%
206   \ifx\x\KVS@empty
207     \toks@{#1}%
208   \else
209     \toks@\expandafter{\the\toks@,#1}%
210   \fi
211   \toks2{#2}%
212   \edef\x{\the\toks2}%
213   \ifx\x\KVS@empty
214     \else
215       \@ReturnAfterFi{%
216       \KVS@@spaceB#2\KVS@nil
217       }%
218   \fi
219 }

```

\KVS@spaceC Removes one space after the comma.

```

220 \def\KVS@spaceC{%
221   \toks@\expandafter{\expandafter}\expandafter
222   \KVS@@spaceC\the\toks@, \KVS@nil
223 }
224 \def\KVS@@spaceC#1, #2\KVS@nil{%
225   \edef\x{\the\toks@}%
226   \ifx\x\KVS@empty
227     \toks@{#1}%

```

```

228 \else
229   \toks@\expandafter{\the\toks@,#1}%
230 \fi
231 \toks2{#2}%
232 \edef\x{\the\toks2}%
233 \ifx\x\KVS@empty
234 \else
235   \@ReturnAfterFi{%
236     \KVS@@spaceC#2\KVS@nil
237   }%
238 \fi
239 }

```

`\KVS@spaceD` Removes one space before the equal sign.

```

240 \def\KVS@spaceD#1{%
241   \toks@\expandafter{\expandafter}\expandafter
242   \KVS@@spaceD\the\toks@#1=\KVS@nil
243 }
244 \def\KVS@@spaceD#1 =#2\KVS@nil{%
245   \edef\x{\the\toks@}%
246   \ifx\x\KVS@empty
247     \toks@{#1}%
248   \else
249     \toks@\expandafter{\the\toks@=#1}%
250   \fi
251   \toks2{#2}%
252   \edef\x{\the\toks2}%
253   \ifx\x\KVS@empty
254   \else
255     \@ReturnAfterFi{%
256       \KVS@@spaceD#2\KVS@nil
257     }%
258   \fi
259 }

```

3.3 Parsing key value lists

`\kv@parse` Normalizes and parses the key value list. Also sets `\kv@list`.

```

260 \def\kv@parse#1{%
261   \kv@normalize{#1}%
262   \expandafter\kv@parse@normalized\expandafter{\kv@list}%
263 }

```

`\kv@parse@normalized`

```

264 \def\kv@parse@normalized#1#2{%
265   \KVS@parse#1,\KVS@nil{#2}%
266 }
267 \def\KVS@parse#1,#2\KVS@nil#3{%
268   \begingroup
269     \toks@{#1}%
270     \edef\x{\the\toks@}%
271     \expandafter\endgroup
272     \ifx\x\KVS@empty
273     \else
274       \KVS@process#1=\KVS@nil{#3}%
275     \fi
276   \begingroup
277     \toks@{#2}%
278     \edef\x{\the\toks@}%
279     \expandafter\endgroup
280     \ifx\x\KVS@empty

```



```

281 \else
282   \@ReturnAfterFi{%
283     \KVS@parse#2\KVS@nil{#3}%
284   }%
285 \fi
286 }

287 \def\KVS@process#1=#2\KVS@nil#3{%
288   \def\kv@key{#1}%
289   \begingroup
290     \toks@{#2}%
291     \edef\x{\the\toks@}%
292   \expandafter\endgroup
293   \ifx\x\KVS@empty
294     \let\kv@value\relax
295     #3{#1}{}%
296   \else
297     \KVS@@process{#1}#2\KVS@nil{#3}%
298   \fi
299 }

300 \def\KVS@@process#1#2=\KVS@nil#3{%
301   \begingroup
302     \toks@{#2}%
303     \xdef\KVS@global{\the\toks@}%
304   \endgroup
305   \let\kv@value\KVS@global
306   #3{#1}{#2}%
307 }

```

3.4 Processing key value pairs

\kv@processor@default

```

308 \def\kv@processor@default#1#2#3{%
309   \begingroup\expandafter\expandafter\expandafter\endgroup
310   \expandafter\ifx\csname KV@#1@#2\endcsname\relax
311     \begingroup\expandafter\expandafter\expandafter\endgroup
312     \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
313       \kv@error@unknownkey{#1}{#2}%
314     \else
315       \csname KVS@#1@handler\endcsname{#2}{#3}%
316     \relax
317   \fi
318 \else
319   \ifx\kv@value\relax
320     \begingroup\expandafter\expandafter\expandafter\endgroup
321     \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
322       \kv@error@novalue{#1}{#2}%
323     \else
324       \csname KV@#1@#2@default\endcsname
325     \relax
326   \fi
327 \else
328   \csname KV@#1@#2\endcsname{#3}%
329 \fi
330 \fi
331 }

```

\kv@set@family@handler

```

332 \def\kv@set@family@handler#1{%
333   \KVS@set@family@handler{#1}\@nil
334 }
335 \def\KVS@set@family@handler#1\@nil#{%

```

```

336 \expandafter\def\csname KVS@#1@handler\endcsname##1##2%
337 }

```

3.5 Error handling

`\kv@error@novalue` Only a poor `\PackageError` is provided by `miniltx.tex`.

```

338 \expandafter\ifx\csname MessageBreak\endcsname\relax
339 \def\MessageBreak{^^J}%
340 \fi
341 \expandafter\ifx\csname @ehc\endcsname\relax
342 \def\@ehc{%
343   Try typing \space\string<return\string> %
344   \space to proceed.\MessageBreak
345   If that doesn't work, type \space X %
346   \string<return\string> \space to quit\string.%
347 }%
348 \fi
349 \def\kv@error@novalue{%
350 \kv@error@generic{No value specified for}%
351 }
352 \def\kv@error@unknownkey{%
353 \kv@error@generic{Undefined}%
354 }
355 \def\kv@error@generic#1#2#3{%
356 \begingroup
357 \newlinechar=10 %
358 \def\MessageBreak{^^J}%
359 \expandafter\ifx\csname PackageError\endcsname\relax
360 \edef\x{%
361 \errhelp{%
362   The keyval family of the key '#3' is '#2'.\MessageBreak
363 \MessageBreak
364 \@ehc
365 }%
366 }%
367 \x
368 \errmessage{kvsetkeys: #1 key '#3'}%
369 \else
370 \edef\x{%
371 \noexpand\PackageError{kvsetkeys}{%
372   #1 key '#3'%
373 }{%
374   The keyval family of the key '#3' is '#2'.\MessageBreak
375 \MessageBreak
376 \@ehc
377 }%
378 }%
379 \x
380 \fi
381 \endgroup
382 }%

```

3.6 Do it all

`\kvsetkeys`

```

383 \def\kvsetkeys#1#2{%
384 \kv@parse{#2}{\kv@processor@default{#1}}%
385 }

386 \KVS@endinput
387 \endpackage

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/kvsetkeys.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/kvsetkeys.pdf](#) Documentation.

Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

[CTAN:macros/latex/contrib/oberdiek/oberdiek-tds.zip](#)

4.2 Bundle installation

Unpacking. Unpack the `oberdiek-tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek-tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting docstrip archive. The files are extracted by running the `.dtx` through plain- \TeX :

```
tex kvsetkeys.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

<code>kvsetkeys.sty</code>	→ <code>tex/generic/oberdiek/kvsetkeys.sty</code>
<code>kvsetkeys.pdf</code>	→ <code>doc/latex/oberdiek/kvsetkeys.pdf</code>
<code>kvsetkeys-example.tex</code>	→ <code>doc/latex/oberdiek/kvsetkeys-example.tex</code>
<code>kvsetkeys.dtx</code>	→ <code>source/latex/oberdiek/kvsetkeys.dtx</code>

If you have a `docstrip.cfg` that configures and enables docstrip’s TDS installing feature, then some files can already be in the right place, see the documentation of docstrip.

4.4 Refresh file name databases

If your \TeX distribution (te \TeX , mi \TeX , ...) relies on file name databases, you must refresh these. For example, te \TeX users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk kvsetkeys.pdf unpack_files output .
```

¹[ftp://ftp.ctan.org/tex-archive/](http://ftp.ctan.org/tex-archive/)

Unpacking with L^AT_EX. The .dtx chooses its action depending on the format:

plain-T_EX: Run docstrip and extract the files.

L^AT_EX: Generate the documentation.

If you insist on using L^AT_EX for docstrip (really, docstrip does not need L^AT_EX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{kvsetkeys.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the .dtx or the .drv to generate the documentation. The process can be configured by the configuration file ltxdoc.cfg. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL^AT_EX:

```
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
```

5 References

- [1] David Carlisle: *The keyval package*; 1999/03/16 v1.13; [CTAN:macros/latex/required/graphics/keyval.dtx](#).

6 History

[2006/03/06 v1.0]

- First version.

[2006/10/19 v1.1]

- Fix of \kv@set@family@handler.
- Example added.

7 Index

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

Symbols	
\,	136, 137
\=	36, 156, 157
\>	38, 39, 40, 41, 42, 43
\@ReturnAfterFi	121,
149, 174, 195, 215, 235, 255, 282	
\@ehc	342, 364, 376
\@empty	13
\@endslash	13, 16, 30
\@nil	333, 335
\ \	37, 38, 39, 40, 41, 42, 43, 44
\~	137, 157
B	
\begin	34, 35
C	
\catcode	50, 51, 52, 53, 54, 78,
79, 80, 81, 82, 83, 84, 85, 103,	

104, 105, 106, 107, 108, 109, 113, 114, 115, 116, 117, 118, 119	\csname 55, 65, 86, 99, 102, 310, 312, 315, 321, 324, 328, 336, 338, 341, 359
D	
\define@key	29
\documentclass	2
\dots	38, 41, 43
E	
\empty	59
\end	45, 46
\endcsname	55, 65, 86, 99, 102, 310, 312, 315, 321, 324, 328, 336, 338, 341, 359
\endinput	74, 111
\errhelp	361
\errmessage	368
I	
\ifcase	56
\ifx ...	57, 59, 65, 86, 94, 147, 165, 172, 186, 193, 206, 213, 226, 233, 246, 253, 272, 280, 293, 310, 312, 319, 321, 338, 341, 359
\immediate	67, 88
K	
\kill	36
\kv@error@generic	350, 353, 355
\kv@error@novalue	322, 338
\kv@error@unknownkey	313, 352
\kv@global	131, 133
\kv@key	288
\kv@list	133, 262
\kv@normalize	2, 122, 261
\kv@parse	2, 260, 384
\kv@parse@normalized	2, 262, 264
\kv@processor@default ...	3, 308, 384
\kv@set@family@handler ...	3, 20, 332
\kv@value	294, 305, 319
\KVS@@comma	141, 143, 150
\KVS@@equal	161, 163, 175
\KVS@@process	297, 300
\KVS@@spaceA	182, 184, 196
\KVS@@spaceB	202, 204, 216
\KVS@@spaceC	222, 224, 236
\KVS@@spaceD	242, 244, 256
\KVS@comma	125, 135
\KVS@empty	120, 147, 165, 172, 186, 193, 206, 213, 226, 233, 246, 253, 272, 280, 293
\KVS@endinput	386
\KVS@equal	126, 155
\KVS@global	303, 305
\KVS@nil 141, 143, 150, 161, 163, 175, 182, 184, 196, 202, 204, 216, 222, 224, 236, 242, 244, 256, 265, 267, 274, 283, 287, 297, 300	
\KVS@parse	265, 267, 283
\KVS@process	274, 287
\KVS@set@family@handler ...	333, 335
\KVS@spaceA	127, 180
\KVS@spaceB	128, 200
\KVS@spaceC	129, 220
\KVS@spaceD	130, 240
\kvsetkeys	3, 14, 383
L	
\lccode	136, 137, 156, 157
\lowercase	138, 158
M	
\makeatletter	7
\makeatother	32
\mbox	36
\MessageBreak	339, 344, 358, 362, 363, 374, 375
N	
\newcommand	8
\newlinechar	357
P	
\PackageError	371
\PackageInfo	70
\ProvidesPackage	100
Q	
\qqquad	36
S	
\space	25, 343, 344, 345, 346
T	
\tag	8, 37, 39, 40, 42, 44
\textgreater	16
\textless	16
\texttt	15
\the	16, 24, 103, 104, 105, 106, 107, 108, 109, 131, 141, 144, 146, 161, 164, 168, 171, 182, 185, 189, 192, 202, 205, 209, 212, 222, 225, 229, 232, 242, 245, 249, 252, 270, 278, 291, 303
\toks	145, 146, 170, 171, 191, 192, 211, 212, 231, 232, 251, 252
\toks@	12, 16, 23, 24, 124, 131, 140, 141, 144, 160, 161, 164, 166, 168, 181, 182, 185, 187, 189, 201, 202, 205, 207, 209, 221, 222, 225, 227, 229, 241, 242, 245, 247, 249, 269, 270, 277, 278, 290, 291, 302, 303
U	
\usepackage	3, 4, 5
W	
\write	67, 88
X	
\x	55, 57, 59, 66, 70, 72, 87, 92, 99, 146, 147, 164, 165, 171, 172, 185, 186, 192, 193, 205, 206, 212, 213, 225, 226, 232, 233, 245, 246, 252, 253, 270, 272, 278, 280, 291, 293, 360, 367, 370, 379