

The kvsetkeys package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2009/12/12 v1.6

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	2
1.1	Motivation	2
1.2	Normalizing key value lists	2
1.3	Parsing key value lists	3
1.4	Processing key value pairs	4
1.5	Default family handler	4
1.6	Put it all together	4
1.7	Comma separated lists	4
2	Example	5
3	Implementation	6
3.1	Identification	6
3.2	Package loading	7
3.3	Check for ε -TeX	7
3.4	Generic help macros	8
3.5	Normalizing	8
3.6	Parsing key value lists	11
3.7	Parsing comma lists	12
3.8	Processing key value pairs	12
3.9	Error handling	13
3.10	Do it all	13
4	Test	13
4.1	Catcode checks for loading	13
4.2	Macro tests	15
4.2.1	Preamble	15
4.2.2	Time	15
4.2.3	Test sets	16
5	Installation	19
5.1	Download	19
5.2	Bundle installation	19
5.3	Package installation	19
5.4	Refresh file name databases	20
5.5	Some details for the interested	20

6	References	20
7	History	20
	[2006/03/06 v1.0]	20
	[2006/10/19 v1.1]	20
	[2007/09/09 v1.2]	21
	[2007/09/29 v1.3]	21
	[2009/07/19 v1.4]	21
	[2009/07/30 v1.5]	21
	[2009/12/12 v1.6]	21
8	Index	21

1 Documentation

First I want to recommend the very good review article “A guide to key-value methods” by Joseph Wright [1]. It introduces the different key-value packages and compares them.

1.1 Motivation

`\kvsetkeys` serves as replacement for `keyval`’s `\setkeys`. It basically uses the same syntax. But the implementation is more robust and predictable:

Active syntax characters: Comma ‘,’ and the equals sign ‘=’ are used inside key value lists as syntax characters. Package `keyval` uses the catcode of the characters that is active during package loading, usually this is catcode 12 (other). But it can happen that the catcode setting of the syntax characters changes. Especially active characters are of interest, because some language adaptations uses them. For example, option `turkish` of package `babel` uses the equals sign as active shorthand character. Therefore package `kvsetkeys` deals with both catcode settings 12 (other) and 13 (active).

Brace removal: Package `keyval`’s `\setkeys` removes up to two levels of curly braces around the value in some unpredictable way:

```
\setkeys{fam}{key={{value}}}  
\setkeys{fam}{key={{value}}}  
\setkeys{fam}{key= {{value}}}
```

This package `kvsetkeys` follows a much stronger rule: Exactly one level of braces are removed from an item, if the item is surrounded by curly braces. An item can be a the key value pair, the key or the value.

```
\kvsetkeys{fam}{key={value}}  
\kvsetkeys{fam}{key={{value}}}  
\kvsetkeys{fam}{key= {{value}}}
```

Arbitrary values: Unmatched conditionals are supported.

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.

1.2 Normalizing key value lists

`\kv@normalize {<key value list>}`

If the user specifies key value lists, he usually prefers 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={...},...,
```

Curly braces around values (or keys) remain untouched.

v1.3+: One comma is added in front of the list and each pair ends with a comma. Thus an empty list consists of one comma, otherwise two commas encloses the list. Empty entries other than the first are removed.

v1.0 – v1.2: 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.

1.3 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>}

```

1.4 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
```

1.5 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`.

1.6 Put it all together

`\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
```

1.7 Comma separated lists

Since version 2007/09/29 v1.3 this package also supports the normalizing and parsing of general comma separated lists.

`\comma@normalize {⟨comma list⟩}`

Macro `\comma@normalize` normalizes the comma separated list, removes spaces around commas. The result is put in macro `\comma@list`.

`\comma@parse {⟨comma list⟩} {⟨processor⟩}`

Macro `\comma@parse` first normalizes the comma separated list and then parses the list by calling `\comma@parse@normalized`.

`\comma@parse@normalized {⟨normalized comma list⟩} {⟨processor⟩}`

The list is parsed. Empty entries are ignored. `⟨processor⟩` is called for each non-empty entry with the entry as argument:

`⟨processor⟩{⟨entry⟩}`

Also the entry is stored in the macro `\comma@entry`.

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} \backslash
38   \> \dots \backslash
39   \> \tag[border=1]{table} \backslash
40   \> \> \tag[width=200, span=3, /]{colgroup} \backslash
41   \> \> \dots \backslash
42   \> \tag{/table} \backslash

```

```

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 \catcode123 1 % {
56 \catcode125 2 % }
57 \expandafter\let\expandafter\x\csname ver@kvsetkeys.sty\endcsname
58 \ifx\x\relax % plain-TeX, first loading
59 \else
60 \def\empty{}%
61 \ifx\x\empty % LaTeX, first loading,
62 % variable is initialized, but \ProvidesPackage not yet seen
63 \else
64 \catcode35 6 % #
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 \aftergroup\endinput
74 \fi
75 \fi
76 \endgroup

```

Package identification:

```

77 \begingroup
78 \catcode35 6 % #
79 \catcode40 12 % (
80 \catcode41 12 % )
81 \catcode44 12 % ,
82 \catcode45 12 % -
83 \catcode46 12 % .
84 \catcode47 12 % /
85 \catcode58 12 % :
86 \catcode64 11 % @
87 \catcode91 12 % [
88 \catcode93 12 % ]
89 \catcode123 1 % {
90 \catcode125 2 % }
91 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
92 \def\x#1#2#3[#4]{\endgroup
93 \immediate\write-1{Package: #3 #4}%
94 \xdef#1{#4}%
95 }%
96 \else

```

```

97     \def\x#1#2[#3]{\endgroup
98     #2[#3]}%
99     \ifx#1\@undefined
100     \xdef#1{#3}%
101     \fi
102     \ifx#1\relax
103     \xdef#1{#3}%
104     \fi
105     }%
106 \fi
107 \expandafter\x\csname ver@kvsetkeys.sty\endcsname
108 \ProvidesPackage{kvsetkeys}%
109 [2009/12/12 v1.6 Key value parser (HO)]
110 \begingroup
111 \catcode123 1 % {
112 \catcode125 2 % }
113 \def\x{\endgroup
114 \expandafter\edef\csname KVS@AtEnd\endcsname{%
115 \catcode35 \the\catcode35\relax
116 \catcode64 \the\catcode64\relax
117 \catcode123 \the\catcode123\relax
118 \catcode125 \the\catcode125\relax
119 }%
120 }%
121 \x
122 \catcode35 6 % #
123 \catcode64 11 % @
124 \catcode123 1 % {
125 \catcode125 2 % }
126 \def\TMP@EnsureCode#1#2{%
127 \edef\KVS@AtEnd{%
128 \KVS@AtEnd
129 \catcode#1 \the\catcode#1\relax
130 }%
131 \catcode#1 #2\relax
132 }
133 \TMP@EnsureCode{36}{3}% $
134 \TMP@EnsureCode{38}{4}% &
135 \TMP@EnsureCode{39}{12}% '
136 \TMP@EnsureCode{44}{12}% ,
137 \TMP@EnsureCode{46}{12}% .
138 \TMP@EnsureCode{47}{12}% /
139 \TMP@EnsureCode{61}{12}% =
140 \TMP@EnsureCode{94}{7}% ^ (superscript)
141 \TMP@EnsureCode{96}{12}% '
142 \TMP@EnsureCode{126}{13}% ~ (active)

```

3.2 Package loading

```

143 \begingroup\expandafter\expandafter\expandafter\endgroup
144 \expandafter\ifx\csname RequirePackage\endcsname\relax
145 \input infwarerr.sty\relax
146 \input etexcmds.sty\relax
147 \else
148 \RequirePackage{infwarerr}[2007/09/09]%
149 \RequirePackage{etexcmds}[2007/09/09]%
150 \fi

```

3.3 Check for ε -TeX

`\unexpanded`, `\ifcsname`, and `\unless` are used if found.

```

151 \begingroup\expandafter\endgroup
152 \ifcase0\ifetex\unexpanded
153 \expandafter\ifx\csname ifcsname\endcsname\relax

```

```

154         \else
155         \expandafter\ifx\csname unless\endcsname\relax
156         \else
157         1%
158         \fi
159         \fi
160         \fi
161     \catcode'\$=9 % ignore
162     \catcode'\&=14 % comment
163 \else % e-TeX
164     \catcode'\$=14 % comment
165     \catcode'\&=9 % ignore
166 \fi

```

3.4 Generic help macros

```

\KVS@Empty
167 \def\KVS@Empty{}

\KVS@FirstOfTwo
168 \long\def\KVS@FirstOfTwo#1#2{#1}

\KVS@SecondOfTwo
169 \long\def\KVS@SecondOfTwo#1#2{#2}

\KVS@IfEmpty
170 \def\KVS@IfEmpty#1{%
171 & \edef\KVS@Temp{\etex@unexpanded{#1}}%
172 $ \begingroup
173 $ \toks@{#1}%
174 $ \edef\KVS@Temp{\the\toks@}%
175 $ \expandafter\endgroup
176 \ifx\KVS@Temp\KVS@Empty
177 \expandafter\KVS@FirstOfTwo
178 \else
179 \expandafter\KVS@SecondOfTwo
180 \fi
181 }

```

3.5 Normalizing

```

\kv@normalize
182 \def\kv@normalize#1{%
183 \begingroup
184 \toks@{, #1,}%
185 \KVS@Comma
186 \KVS@SpaceComma{ }%
187 \KVS@CommaSpace
188 \KVS@CommaComma
189 \KVS@Equals
190 \KVS@SpaceEquals{ }%
191 \KVS@EqualsSpace
192 \xdef\KVS@Global{\the\toks@}%
193 \endgroup
194 \let\kv@list\KVS@Global
195 }

\comma@normalize
196 \def\comma@normalize#1{%
197 \begingroup

```



```

198 \toks@{, #1,}%
199 \KVS@Comma
200 \KVS@SpaceComma{ }%
201 \KVS@CommaSpace
202 \KVS@CommaComma
203 \xdef\KVS@Global{\the\toks@}%
204 \endgroup
205 \let\comma@list\KVS@Global
206 }

```

`\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.

```

207 \begingroup
208 \lccode'\,='\",%
209 \lccode'\~='\",%
210 \lowercase{\endgroup
211 \def\KVS@Comma{%
212 \toks@\expandafter{\expandafter}\expandafter
213 \KVS@@Comma\the\toks@\KVS@Nil
214 }%
215 \def\KVS@@Comma#1~#2\KVS@Nil{%
216 \toks@\expandafter{\the\toks@#1}%
217 \KVS@IfEmpty{#2}{%
218 }{%
219 \KVS@@Comma,#2\KVS@Nil
220 }%
221 }%
222 }

```

`\KVS@SpaceComma` Removes spaces before the comma, may add commas at the end.

```

223 \def\KVS@SpaceComma#1{%
224 \toks@\expandafter{\the\toks@#1,}%
225 \expandafter\KVS@@SpaceComma\the\toks@\KVS@Nil
226 }

```

`\KVS@@SpaceComma`

```

227 \def\KVS@@SpaceComma#1 ,#2\KVS@Nil{%
228 \KVS@IfEmpty{#2}{%
229 \toks@{#1}%
230 }{%
231 \toks@{#1,#2}%
232 \expandafter\KVS@@SpaceComma\the\toks@\KVS@Nil
233 }%
234 }

```

`\KVS@CommaSpace` Removes spaces after the comma, may add commas at the end.

```

235 \def\KVS@CommaSpace{%
236 \toks@\expandafter{\the\toks@, }%
237 \expandafter\KVS@@CommaSpace\the\toks@\KVS@Nil
238 }

```

`\KVS@@CommaSpace`

```

239 \def\KVS@@CommaSpace#1, #2\KVS@Nil{%
240 \KVS@IfEmpty{#2}{%
241 \toks@{#1}%
242 }{%
243 \toks@{#1,#2}%
244 \expandafter\KVS@@CommaSpace\the\toks@\KVS@Nil
245 }%
246 }

```

`\KVS@CommaComma` Replaces multiple commas by one comma.

```
247 \def\KVS@CommaComma{%
248   \toks@\expandafter{\the\toks@,}%
249   \expandafter\KVS@@CommaComma\the\toks@\KVS@Nil
250 }
```

`\KVS@@CommaComma`

```
251 \def\KVS@@CommaComma#1,,#2\KVS@Nil{%
252   \toks@{#1,#2}%
253   \KVS@IfEmpty{#2}{%
254     }{%
255     \expandafter\KVS@@CommaComma\the\toks@\KVS@Nil
256   }%
257 }
```

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

```
258 \begingroup
259   \lccode'\=='\=%
260   \lccode'\~='\=%
261 \lowercase{\endgroup
262   \def\KVS@Equals{%
263     \toks@\expandafter{\expandafter}\expandafter
264     \KVS@@Equals\the\toks@\KVS@Nil
265   }%
266   \def\KVS@@Equals#1~#2\KVS@Nil{%
267     \edef\KVS@Temp{\the\toks@}%
268     \ifx\KVS@Temp\KVS@Empty
269       \expandafter\KVS@FirstOfTwo
270     \else
271       \expandafter\KVS@SecondOfTwo
272     \fi
273     {%
274       \toks@{#1}%
275     }{%
276       \toks@\expandafter{\the\toks@=#1}%
277     }%
278     \KVS@IfEmpty{#2}{%
279     }{%
280       \KVS@@Equals#2\KVS@Nil
281     }%
282   }%
283 }
```

`\KVS@SpaceEquals` Removes spaces before the equals sign.

```
284 \def\KVS@SpaceEquals#1{%
285   \toks@\expandafter{\the\toks@#1}%
286   \expandafter\KVS@@SpaceEquals\the\toks@\KVS@Nil
287 }
```

`\KVS@@SpaceEquals`

```
288 \def\KVS@@SpaceEquals#1=#2\KVS@Nil{%
289   \KVS@IfEmpty{#2}{%
290     \toks@{#1}%
291   }{%
292     \toks@{#1=#2}%
293     \expandafter\KVS@@SpaceEquals\the\toks@\KVS@Nil
294   }%
295 }
```

`\KVS@EqualsSpace` Removes spaces after the equals sign.

```
296 \def\KVS@EqualsSpace{%
```

```

297 \toks@\expandafter{\the\toks@= }%
298 \expandafter\KVS@@EqualsSpace\the\toks@\KVS@Nil
299 }

```

\KVS@@EqualsSpace

```

300 \def\KVS@@EqualsSpace#1= #2\KVS@Nil{%
301   \KVS@IfEmpty{#2}{%
302     \toks@{#1}%
303   }{%
304     \toks@{#1=#2}%
305   \expandafter\KVS@@EqualsSpace\the\toks@\KVS@Nil
306   }%
307 }

```

3.6 Parsing key value lists

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

```

308 \def\kv@parse#1{%
309   \kv@normalize{#1}%
310   \expandafter\kv@parse@normalized\expandafter{\kv@list}%
311 }

```

\kv@parse@normalized #1: key value list

#2: processor

```

312 \def\kv@parse@normalized#1#2{%
313   \KVS@Parse#1,\KVS@Nil{#2}%
314 }

```

\KVS@Parse #1,#2: key value list

#3: processor

```

315 \def\KVS@Parse#1,#2\KVS@Nil#3{%
316   \KVS@IfEmpty{#1}{%
317   }{%
318     \KVS@Process#1=\KVS@Nil{#3}%
319   }%
320   \KVS@IfEmpty{#2}{%
321   }{%
322     \KVS@Parse#2\KVS@Nil{#3}%
323   }%
324 }

```

\KVS@Process #1: key

#2: value, =

#3: processor

```

325 \def\KVS@Process#1=#2\KVS@Nil#3{%
326   \def\kv@key{#1}%
327   \KVS@IfEmpty{#2}{%
328     \let\kv@value\relax
329     #3{#1}{}%
330   }{%
331     \KVS@@Process{#1}#2\KVS@Nil{#3}%
332   }%
333 }

```

\KVS@@Process #1: key

#2: value

#3: processor

```

334 \def\KVS@@Process#1#2=\KVS@Nil#3{%
335   & \edef\kv@value{\etex@unexpanded{#2}}%
336   $ \begingroup
337   $ \toks@{#2}%

```

```

338 $ \xdef\KVS@Global{\the\toks@}%
339 $ \endgroup
340 $ \let\kv@value\KVS@Global
341 #3{#1}{#2}%
342 }

```

3.7 Parsing comma lists

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

```

343 \def\comma@parse#1{%
344   \comma@normalize{#1}%
345   \expandafter\comma@parse@normalized\expandafter{\comma@list}%
346 }

```

```

\comma@parse@normalized #1: comma list
                        #2: processor
347 \def\comma@parse@normalized#1#2{%
348   \KVS@CommaParse#1,\KVS@Nil{#2}%
349 }

```

```

\KVS@CommaParse #1,#2: comma list
                 #3: processor
350 \def\KVS@CommaParse#1,#2\KVS@Nil#3{%
351   \KVS@IfEmpty{#1}{%
352     }{%
353     \def\comma@entry{#1}%
354     #3{#1}%
355     }%
356   \KVS@IfEmpty{#2}{%
357     }{%
358     \KVS@CommaParse#2\KVS@Nil{#3}%
359     }%
360 }

```

3.8 Processing key value pairs

```

\kv@processor@default
361 \def\kv@processor@default#1#2#3{%
362   & \unless\ifcsname KV@#1@#2\endcsname
363   $ \begingroup\expandafter\expandafter\expandafter\endgroup
364   $ \expandafter\ifx\csname KV@#1@#2\endcsname\relax
365   & \unless\ifcsname KVS@#1@handler\endcsname
366   $ \begingroup\expandafter\expandafter\expandafter\endgroup
367   $ \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
368     \kv@error@unknownkey{#1}{#2}%
369   \else
370     \csname KVS@#1@handler\endcsname{#2}{#3}%
371     \relax
372   \fi
373   \else
374     \ifx\kv@value\relax
375     & \unless\ifcsname KV@#1@#2@default\endcsname
376     $ \begingroup\expandafter\expandafter\expandafter\endgroup
377     $ \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
378       \kv@error@novalue{#1}{#2}%
379     \else
380       \csname KV@#1@#2@default\endcsname
381       \relax
382     \fi
383   \else
384     \csname KV@#1@#2\endcsname{#3}%

```

```

385     \fi
386 \fi
387 }

```

`\kv@set@family@handler`

```

388 \def\kv@set@family@handler#1{%
389   \KVS@SetFamilyHandler{#1}\@nil
390 }

```

`\KVS@SetFamilyHandler`

```

391 \def\KVS@SetFamilyHandler#1\@nil#{%
392   \expandafter\def\csname KVS@#1@handler\endcsname##1##2%
393 }

```

3.9 Error handling

`\kv@error@novalue`

```

394 \def\kv@error@novalue{%
395   \kv@error@generic{No value specified for}%
396 }

```

`\kv@error@unknownkey`

```

397 \def\kv@error@unknownkey{%
398   \kv@error@generic{Undefined}%
399 }

```

`\kv@error@generic`

```

400 \def\kv@error@generic#1#2#3{%
401   \@PackageError{kvsetkeys}{%
402     #1 key ‘#3’%
403   }{%
404     The keyval family of the key ‘#3’ is ‘#2’.\MessageBreak
405     \MessageBreak
406     \@ehc
407   }%
408 }

```

3.10 Do it all

`\kvsetkeys`

```

409 \def\kvsetkeys#1#2{%
410   \kv@parse{#2}{\kv@processor@default{#1}}%
411 }

412 \KVS@AtEnd
413 \</package>

```

4 Test

4.1 Catcode checks for loading

```

414 \<test1>

415 \catcode‘\{=1 %
416 \catcode‘\}=2 %
417 \catcode‘\#=6 %
418 \catcode‘\@=11 %
419 \expandafter\ifx\csname count@\endcsname\relax
420   \countdef\count@=255 %
421 \fi

```

```

422 \expandafter\ifx\csname @gobble\endcsname\relax
423   \long\def\@gobble#1{}%
424 \fi
425 \expandafter\ifx\csname @firstofone\endcsname\relax
426   \long\def\@firstofone#1{#1}%
427 \fi
428 \expandafter\ifx\csname loop\endcsname\relax
429   \expandafter\@firstofone
430 \else
431   \expandafter\@gobble
432 \fi
433 {%
434   \def\loop#1\repeat{%
435     \def\body{#1}%
436     \iterate
437   }%
438   \def\iterate{%
439     \body
440     \let\next\iterate
441   \else
442     \let\next\relax
443   \fi
444   \next
445 }%
446 \let\repeat=\fi
447 }%
448 \def\RestoreCatcodes{}
449 \count@=0 %
450 \loop
451   \edef\RestoreCatcodes{%
452     \RestoreCatcodes
453     \catcode\the\count@=\the\catcode\count@\relax
454   }%
455 \ifnum\count@<255 %
456   \advance\count@ 1 %
457 \repeat
458
459 \def\RangeCatcodeInvalid#1#2{%
460   \count@=#1\relax
461   \loop
462     \catcode\count@=15 %
463   \ifnum\count@<#2\relax
464     \advance\count@ 1 %
465   \repeat
466 }
467 \expandafter\ifx\csname LoadCommand\endcsname\relax
468   \def\LoadCommand{\input kvsetkeys.sty\relax}%
469 \fi
470 \def\Test{%
471   \RangeCatcodeInvalid{0}{47}%
472   \RangeCatcodeInvalid{58}{64}%
473   \RangeCatcodeInvalid{91}{96}%
474   \RangeCatcodeInvalid{123}{255}%
475   \catcode'\@=12 %
476   \catcode'\=0 %
477   \catcode'\{=1 %
478   \catcode'\}=2 %
479   \catcode'\#=6 %
480   \catcode'\[=12 %
481   \catcode'\]=12 %
482   \catcode'\%=14 %
483   \catcode'\ =10 %

```

```

484 \catcode13=5 %
485 \LoadCommand
486 \RestoreCatcodes
487 }
488 \Test
489 \csname @@end\endcsname
490 \end
491 \test1\

```

4.2 Macro tests

4.2.1 Preamble

```

492 (*test2\
493 \NeedsTeXFormat{LaTeX2e}
494 \nofiles
495 \documentclass{article}
496 (noetex)\let\SavedUnexpanded\unexpanded
497 (noetex)\let\unexpanded\UNDEFINED
498 \makeatletter
499 \chardef\KVS@TestMode=1 %
500 \makeatother
501 \usepackage{kvsetkeys}[2009/12/12]
502 (noetex)\let\unexpanded\SavedUnexpanded
503 \usepackage{qstest}
504 \IncludeTests{*}
505 \LogTests{log}{*}{*}

```

4.2.2 Time

```

506 \begingroup\expandafter\expandafter\expandafter\endgroup
507 \expandafter\ifx\csname pdfresettimer\endcsname\relax
508 \else
509   \makeatletter
510   \newcount\SummaryTime
511   \newcount\TestTime
512   \SummaryTime=\z@
513   \newcommand*{\PrintTime}[2]{%
514     \typeout{%
515       [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
516     }%
517   }%
518   \newcommand*{\StartTime}[1]{%
519     \renewcommand*{\TimeDescription}{#1}%
520     \pdfresettimer
521   }%
522   \newcommand*{\TimeDescription}{}%
523   \newcommand*{\StopTime}{}%
524   \TestTime=\pdfelapsedtime
525   \global\advance\SummaryTime\TestTime
526   \PrintTime\TimeDescription\TestTime
527   }%
528   \let\saved@qstest\qstest
529   \let\saved@endqstest\endqstest
530   \def\qstest#1#2{%
531     \saved@qstest{#1}{#2}%
532     \StartTime{#1}%
533   }%
534   \def\endqstest{%
535     \StopTime
536     \saved@endqstest
537   }%
538   \AtEndDocument{%
539     \PrintTime{summary}\SummaryTime

```

```

540 }%
541 \makeatother
542 \fi

```

4.2.3 Test sets

```

543 \makeatletter
544 \def\@makeactive#1{%
545   \catcode'#1=13\relax
546 }
547 \@makeactive\,
548 \def,{\errmessage{COMMA}}
549 \@makeoother\,
550 \@makeactive\=
551 \def={\errmessage{EQUALS}}
552 \@makeoother\=
553
554 \begin{qstest}{normalize}{normalize,active-chars,space-removal}%
555   \def\Test#1#2{%
556     \@makeoother\,%
557     \@makeoother\=%
558     \scantokens{\toks@={#2}}%
559     \edef\Result{\the\toks@}%
560     \@makeoother\,%
561     \@makeoother\=%
562     \@Test{#1}%
563     \@makeactive\,%
564     \@Test{#1}%
565     \@makeactive\=%
566     \@Test{#1}%
567     \@makeoother\,%
568     \@Test{#1}%
569     \@makeoother\=%
570   }%
571   \def\@Test#1{%
572     \scantokens{\kv@normalize{#1}}%
573     \expandafter\expandafter\expandafter\Expect
574     \expandafter\expandafter\expandafter
575     {\expandafter\kv@list\expandafter}\expandafter{\Result}%
576     \Expect*{\ifx\kv@list\Result true\else false\fi}{true}%
577   }%
578   \Test{}{},{}%
579   \Test{,}{,}%
580   \Test{,,}{,}%
581   \Test{,,,}{,}%
582   \Test{ , }{,}%
583   \Test{{a}}{,{a},}%
584   \Test{,{a}}{,{a},}%
585   \Test{{a},}{,{a},}%
586   \Test{{a},{b}}{,{a},{b},}%
587   \Test{{b}={c},{}=,{}=,{d}={},{b}={c},{}=,{d}={},{d}={},}%
588   \Test{{}}{,{},}%
589   \Test{{},{},{}}{,{},{},}%
590   \Test{=}{,=,}%
591   \Test{=,=,=}{,=,=,}%
592   \def\TestSet#1{%
593     \Test{#1#1}{,}%
594     \Test{#1#1,#1#1}{,}%
595     \Test{#1#1,#1#1,#1#1}{,}%
596     \Test{#1#1#1#1#1}{,}%
597     \Test{{a}#1#1=#1#1{b}}{,{a}={b},}%
598   }%
599   \TestSet{ }%
600 \begingroup

```



```

601 \let\saved@normalize\kv@normalize
602 \def\kv@normalize#1{%
603   \saved@normalize{#1}%
604   \@onelevel@sanitize\kv@list
605   \@onelevel@sanitize\Result
606 }%
607 \Test{#,#=#,{#}={#},{#}=#,{#}}{,{#,#=#,{#}={#},{#}=#,{#}},}%
608 \endgroup
609 \begin{group}
610   \def\Test#1#2{%
611     \edef\Result{#2}%
612     \@Test{#1}%
613   }%
614   \Test{{ a = b }}{,{ a = b },}%
615   \@makeactive\,%
616   \Test{{,}}{\string,{\noexpand,}\string,}%
617   \@makeother\,%
618   \@makeactive\=%
619   \Test{a={}}{,{a\string={\noexpand=},}%
620 \end{group}
621 \Test{a=b}{,{a=b},}%
622 \Test{a={b}}{,{a={b}},}%
623 \Test{a = {b}}{,{a={b}},}%
624 \Test{a= {b}}{,{a={b}},}%
625 \Test{a = {b}}{,{a={b}},}%
626 \Test{a = {b} ,}{,{a={b}},}%
627 \Test{a}{,{a},}%
628 \Test{ a}{,{a},}%
629 \Test{ a }{,{a},}%
630 \Test{ a }{,{a},}%
631 \Test{, a ,}{,{a},}%
632 \Test{, a b ,}{,{a b},}%
633 \Test{, a ,}{,{a},}%
634 \Test{ a =}{,{a=},}%
635 \Test{ a = }{,{a=},}%
636 \Test{ a = }{,{a=},}%
637 \Test{{a} =}{,{a}=},}%
638 \Test{{a}= {}}{,{a}={}},}%
639 \Test{, a = {}}{,{a=}},}%
640 \Test{a,b}{,{a,b},}%
641 \Test{a=\fi}{,{a=\fi},}%
642 \Test{a=\iffalse}{,{a=\iffalse},}%
643 \Test{a=\iffalse,b=\fi}{,{a=\iffalse,b=\fi},}%
644 \end{qstest}
645
646 \begin{qstest}{\parse}{\parse,brace-removal}
647   \def\Processor#1#2{%
648     \expandafter\Expect\expandafter{\kv@key}{#1}%
649     \toks@{#2}%
650     \edef\x{\the\toks@}%
651     \ifx\kv@value\relax
652       \Expect*{\the\toks@}{}%
653       \def\Value{<>}%
654     \else
655       \edef\Value{[\the\toks@]}%
656       \@onelevel@sanitize\Value
657     \fi
658     \toks@{#1}%
659     \ifx\Result\@empty
660       \edef\Result{[\the\toks@]=\Value}%
661     \else
662       \edef\Result{\Result,[\the\toks@]=\Value}%

```

```

663     \fi
664     \@onelevel@sanitize\Result
665 }%
666 \def\Test#1#2{%
667     \sbox0{%
668         \let\Result\@empty
669         \kv@parse{#1}\Processor
670         \Expect*{\Result}{#2}%
671     }%
672     \Expect*{\the\wd0}{0.0pt}%
673 }%
674 \Test{}{}%
675 \Test{{}}{}%
676 \Test{{{}}}{[]=<>}%
677 \Test{{{}}}{[{}]=<>}%
678 \Test{a}{[a]=<>}%
679 \Test{{a}}{[a]=<>}%
680 \Test{{{a}}}{[a]=<>}%
681 \Test{{{a}}}{[a]=<>}%
682 \Test{{{a}}}{[a]=<>}%
683 \Test{a=}{[a]=[]}%
684 \Test{a=}{[a]=[]}%
685 \Test{{{a}}}{[a]=[]}%
686 \Test{a={}}{[a]=[]}%
687 \Test{a={}}{[a]=[{}]}%
688 \Test{a=b}{[a]=[b]}%
689 \Test{a=\fi}{[a]=[\fi]}%
690 \Test{a=\iffalse}{[a]=[\iffalse]}%
691 \Test{a=\iffalse,b=\fi}{[a]=[\iffalse],[b]=[\fi]}%
692 \Test{{ a = b }}{[ a ]=[ b ]}%
693 \Test{{{ a = b }}}{[ a = b ]=<>}%
694 \end{qstest}
695
696 \begin{qstest}{comma}{comma,parse}
697     \def\Processor#1{%
698         \expandafter\Expect\expandafter{\comma@entry}{#1}%
699         \toks@{#1}%
700         \ifx\Result\@empty
701             \edef\Result{[\the\toks@]}%
702         \else
703             \edef\Result{\Result,[\the\toks@]}%
704         \fi
705         \@onelevel@sanitize\Result
706     }%
707     \def\Test#1#2{%
708         \sbox0{%
709             \let\Result\@empty
710             \comma@parse{#1}\Processor
711             \Expect*{\Result}{#2}%
712         }%
713         \Expect*{\the\wd0}{0.0pt}%
714     }%
715     \Test{}{}%
716     \Test{{}}{}%
717     \Test{{{}}}{[{}]}%
718     \Test{a}{[a]}%
719     \Test{{a}}{[a]}%
720     \Test{{{a}}}{[a]}%
721     \Test{a=}{[a=]}%
722     \Test{a\fi}{[a\fi]}%
723     \Test{a\iffalse}{[a\iffalse]}%
724     \Test{\iffalse,\fi}{[\iffalse],[\fi]}%

```

```

725 \Test{ a , b , c }{[a],[b],[c]}%
726 \Test{ { } ,{ } , { } , { } , { } }{[ ],[ ],[ ],[ ],[ ]}%
727 \Test{ {{}} ,{{}}, {{}}, {{}} , {{}} }{{}},{{}},{{}},{{}},{{}}}%
728 \end{qstest}
729
730 \begin{document}
731 \end{document}
732 \</test2>

```

5 Installation

5.1 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:install/macros/latex/contrib/oberdiek.tds.zip](#)

TDS refers to the standard “A Directory Structure for T_EX Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

5.2 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/
```

5.3 Package installation

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

```
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):

```

kvsetkeys.sty      → tex/generic/oberdiek/kvsetkeys.sty
kvsetkeys.pdf      → doc/latex/oberdiek/kvsetkeys.pdf
kvsetkeys-example.tex → doc/latex/oberdiek/kvsetkeys-example.tex
test/kvsetkeys-test1.tex → doc/latex/oberdiek/test/kvsetkeys-test1.tex
test/kvsetkeys-test2.tex → doc/latex/oberdiek/test/kvsetkeys-test2.tex
test/kvsetkeys-test3.tex → doc/latex/oberdiek/test/kvsetkeys-test3.tex
kvsetkeys.dtx      → source/latex/oberdiek/kvsetkeys.dtx

```

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.

¹<http://ftp.ctan.org/tex-archive/>

5.4 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`.

5.5 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 .
```

Unpacking with \LaTeX . The `.dtx` chooses its action depending on the format:

plain- \TeX : Run `docstrip` and extract the files.

\LaTeX : Generate the documentation.

If you insist on using \LaTeX for `docstrip` (really, `docstrip` does not need \LaTeX), 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 pdf \LaTeX :

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

6 References

- [1] A guide to key-value methods, Joseph Wright, second draft for **TUG-Boat**, 2009-03-17. <http://www.texdev.net/wp-content/uploads/2009/03/keyval.pdf>
- [2] David Carlisle: *The keyval package*; 1999/03/16 v1.13; **CTAN:macros/latex/required/graphics/keyval.dtx**.

7 History

[2006/03/06 v1.0]

- First version.

[2006/10/19 v1.1]

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

[2007/09/09 v1.2]

- Using package `infwarerr` for error messages.
- Catcode section rewritten.

[2007/09/29 v1.3]

- Normalizing and parsing of comma separated lists added.
- `\kv@normalize` rewritten.
- Robustness increased for normalizing and parsing, e.g. for values with unmatched conditionals.
- ε -TeX is used if available.
- Tests added for normalizing and parsing.

[2009/07/19 v1.4]

- Bug fix for `\kv@normalize`: unwanted space removed (Florent Chervet).

[2009/07/30 v1.5]

- Documentation addition: recommendation for Joseph Wright's review article.

[2009/12/12 v1.6]

- Short info shortened.

8 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		\@onelevel@sanitize	
\#	417, 479 604, 605, 656, 664, 705	
\\$	161, 164	\@undefined	
\%	482	\[.....	
\&	162, 165	\\ ... 37, 38, 39, 40, 41, 42, 43, 44, 476	
\,	208, 209, 547, 549, 556, 560, 563, 567, 615, 617	\{	
\=	36, 259, 260, 550, 552, 557, 561, 565, 569, 618	\}	
\>	38, 39, 40, 41, 42, 43	\]	
\@	418, 475	\~	
\@PackageError	401	_	
\@Test	562, 564, 566, 568, 571, 612		
\@ehc	406	A	
\@empty	13, 659, 668, 700, 709	\advance	
\@endslash	13, 16, 30	\aftergroup	
\@firstofone	426, 429	\AtEndDocument	
\@gobble	423, 431		
\@makeactive	544, 547, 550, 563, 565, 615, 618	B	
\@makeoother	549, 552, 556, 557, 560, 561, 567, 569, 617	\begin	
\@nil	389, 391	\body	
		C	
		\catcode	

P	
\PackageInfo	70
\pdfelapsedtime	524
\pdfresettimer	520
\PrintTime	513, 526, 539
\Processor	647, 669, 697, 710
\ProvidesPackage	62, 108
Q	
\qqquad	36
\qstest	528, 530
R	
\RangeCatcodeInvalid	459, 471, 472, 473, 474
\renewcommand	519
\repeat	434, 446, 457, 465
\RequirePackage	148, 149
\RestoreCatcodes ..	448, 451, 452, 486
\Result	559, 575, 576, 605, 611, 659, 660, 662, 664, 668, 670, 700, 701, 703, 705, 709, 711
S	
\saved@endqstest	529, 536
\saved@normalize	601, 603
\saved@qstest	528, 531
\SavedUnexpanded	496, 502
\sbox	667, 708
\scantokens	558, 572
\space	25, 515
\StartTime	518, 532
\StopTime	523, 535
\strip@pt	515
\SummaryTime	510, 512, 525, 539
T	
\tag	8, 37, 39, 40, 42, 44
\Test ..	470, 488, 555, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 593, 594, 595, 596, 597, 607, 610, 614, 616, 619, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 666, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688,
	689, 690, 691, 692, 693, 707, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727
\TestSet	592, 599
\TestTime	511, 524, 525, 526
\textgreater	16
\textless	16
\texttt	15
\the	16, 24, 115, 116, 117, 118, 129, 174, 192, 203, 213, 216, 224, 225, 232, 236, 237, 244, 248, 249, 255, 264, 267, 276, 285, 286, 293, 297, 298, 305, 338, 453, 559, 650, 652, 655, 660, 662, 672, 701, 703, 713
\TimeDescription	519, 522, 526
\TMP@EnsureCode ..	126, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142
\toks@ ..	12, 16, 23, 24, 173, 174, 184, 192, 198, 203, 212, 213, 216, 224, 225, 229, 231, 232, 236, 237, 241, 243, 244, 248, 249, 252, 255, 263, 264, 267, 274, 276, 285, 286, 290, 292, 293, 297, 298, 302, 304, 305, 337, 338, 558, 559, 649, 650, 652, 655, 658, 660, 662, 699, 701, 703
\typeout	514
U	
\UNDEFINED	497
\unexpanded	496, 497, 502
\unless	362, 365, 375
\usepackage	3, 4, 5, 501, 503
V	
\Value	653, 655, 656, 660, 662
W	
\wd	672, 713
\write	67, 93
X	
\x	57, 58, 61, 66, 70, 72, 92, 97, 107, 113, 121, 650
Z	
\z@	512