

# The kvsetkeys package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2007/09/29 v1.3

## 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

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Motivation	2
1.2	Normalizing key value lists	2
1.3	Parsing key value lists	3
1.4	Processing key value pairs	3
1.5	Default family handler	4
1.6	Put it all together	4
1.7	Comma separated lists	4
<b>2</b>	<b>Example</b>	<b>5</b>
<b>3</b>	<b>Implementation</b>	<b>5</b>
3.1	Identification	5
3.2	Package loading	7
3.3	Check for $\varepsilon$ -TeX	7
3.4	Generic help macros	8
3.5	Normalizing	8
3.6	Parsing key value lists	11
3.7	Parsing comma lists	11
3.8	Processing key value pairs	12
3.9	Error handling	13
3.10	Do it all	13
<b>4</b>	<b>Test</b>	<b>13</b>
4.1	Catcode checks for loading	13
4.2	Macro tests	14
4.2.1	Preamble	14
4.2.2	Time	15
4.2.3	Test sets	15
<b>5</b>	<b>Installation</b>	<b>18</b>
5.1	Download	18
5.2	Bundle installation	19
5.3	Package installation	19
5.4	Refresh file name databases	19
5.5	Some details for the interested	19

<b>6</b>	<b>References</b>	<b>20</b>
<b>7</b>	<b>History</b>	<b>20</b>
	[2006/03/06 v1.0]	20
	[2006/10/19 v1.1]	20
	[2007/09/09 v1.2]	20
	[2007/09/29 v1.3]	20
<b>8</b>	<b>Index</b>	<b>21</b>

# 1 Documentation

## 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  $\langle key \rangle$  exists
  call the keyval code of  $\langle key \rangle$ 
else
  if  $\langle handler \rangle$  for  $\langle family \rangle$  exists
     $\langle handler \rangle \{ \langle key \rangle \} \{ \langle value \rangle \}$ 
  else
    raise unknown key error
  fi
fi

```

## 1.5 Default family handler

`\kv@processor@default` calls  $\langle handler \rangle$ , 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 { $\langle family \rangle$ } { $\langle handler definition \rangle$ }
```

This sets the default family handler for the keyval family  $\langle family \rangle$ . Inside  $\langle handler definition \rangle$  #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 { $\langle family \rangle$ } { $\langle key value list \rangle$ }
```

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

```
\kv@parse { $\langle key value list \rangle$ } { \kv@processor@default { $\langle family \rangle$ } }
```

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 { $\langle comma list \rangle$ }
```

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

```
\comma@parse { $\langle comma list \rangle$ } { $\langle processor \rangle$ }
```

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

```
\comma@parse@normalized { $\langle normalized comma list \rangle$ } { $\langle processor \rangle$ }
```

The list is parsed. Empty entries are ignored.  $\langle processor \rangle$  is called for each non-empty entry with the entry as argument:

$\langle processor \rangle \{ \langle entry \rangle \}$

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} \\\
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 \*example)

```

## 3 Implementation

### 3.1 Identification

48 (\*package)

Reload check, especially if the package is not used with L<sup>A</sup>T<sub>E</sub>X.

```
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     \catcode35 6 % #
66     \catcode123 1 % {
67     \catcode125 2 % }
68     \expandafter\ifx\csname PackageInfo\endcsname\relax
69       \def\x#1#2{%
70         \immediate\write-1{Package #1 Info: #2.}%
71       }%
72     \else
73       \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
74     \fi
75     \x{kvsetkeys}{The package is already loaded}%
76   \endgroup
77   \expandafter\endinput
78 \fi
79 \endgroup
```

Package identification:

```
80 \begingroup
81   \catcode35 6 % #
82   \catcode40 12 % (
83   \catcode41 12 % )
84   \catcode44 12 % ,
85   \catcode45 12 % -
86   \catcode46 12 % .
87   \catcode47 12 % /
88   \catcode58 12 % :
89   \catcode64 11 % @
90   \catcode123 1 % {
91   \catcode125 2 % }
92   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
93     \def\x#1#2#3[#4]{\endgroup
94       \immediate\write-1{Package: #3 #4}%
95       \xdef#1{#4}%
96     }%
97   \else
98     \def\x#1#2[#3]{\endgroup
99       #2[#3]}%
100     \ifx#1\relax
101       \xdef#1{#3}%
102     \fi
103   }%
104 \fi
105 \expandafter\x\csname ver@kvsetkeys.sty\endcsname
106 \ProvidesPackage{kvsetkeys}%
107 [2007/09/29 v1.3 Key value parser with default handler support (H0)]
```

```

108 \begingroup
109   \catcode123 1 % {
110   \catcode125 2 % }
111   \def\x{\endgroup
112     \expandafter\edef\csname KVS@AtEnd\endcsname{%
113       \catcode35 \the\catcode35\relax
114       \catcode64 \the\catcode64\relax
115       \catcode123 \the\catcode123\relax
116       \catcode125 \the\catcode125\relax
117     }%
118   }%
119 \x
120 \catcode35 6 % #
121 \catcode64 11 % @
122 \catcode123 1 % {
123 \catcode125 2 % }
124 \def\TMP@EnsureCode#1#2{%
125   \edef\KVS@AtEnd{%
126     \KVS@AtEnd
127     \catcode#1 \the\catcode#1\relax
128   }%
129   \catcode#1 #2\relax
130 }
131 \TMP@EnsureCode{36}{3}% $
132 \TMP@EnsureCode{38}{4}% &
133 \TMP@EnsureCode{39}{12}% '
134 \TMP@EnsureCode{44}{12}% ,
135 \TMP@EnsureCode{46}{12}% .
136 \TMP@EnsureCode{47}{12}% /
137 \TMP@EnsureCode{61}{12}% =
138 \TMP@EnsureCode{94}{7}% ^ (superscript)
139 \TMP@EnsureCode{96}{12}% '
140 \TMP@EnsureCode{126}{13}% ~ (active)

```

### 3.2 Package loading

```

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

```

### 3.3 Check for $\epsilon$ -TeX

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

```

149 \begingroup\expandafter\endgroup
150 \ifcase0\ifetex@unexpanded
151   \expandafter\ifx\csname ifcsname\endcsname\relax
152   \else
153     \expandafter\ifx\csname unless\endcsname\relax
154     \else
155       1%
156     \fi
157   \fi
158 \fi
159 \catcode'\$=9 % ignore
160 \catcode'\&=14 % comment
161 \else % e-TeX
162   \catcode'\$=14 % comment
163   \catcode'\&=9 % ignore
164 \fi

```

### 3.4 Generic help macros

```

\KVS@Empty
165 \def\KVS@Empty{}

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

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

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

```

### 3.5 Normalizing

```

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

\comma@normalize
194 \def\comma@normalize#1{%
195 \begingroup
196 \toks@{, #1,}%
197 \KVS@Comma
198 \KVS@SpaceComma{ }%
199 \KVS@CommaSpace
200 \KVS@CommaComma
201 \xdef\KVS@Global{\the\toks@}%
202 \endgroup
203 \let\comma@list\KVS@Global
204 }

```

**\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.

```

205 \begingroup
206 \lccode'\,=\,%
207 \lccode'\~= '\,%

```



```

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

```

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

```

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

```

\KVS@@SpaceComma

```

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

```

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

```

233 \def\KVS@CommaSpace{%
234   \toks@\expandafter{\the\toks@,}%
235   \expandafter\KVS@@CommaSpace\the\toks@\KVS@Nil
236 }

```

\KVS@@CommaSpace

```

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

```

\KVS@CommaComma Replaces multiple commas by one comma.

```

245 \def\KVS@CommaComma{%
246   \toks@\expandafter{\the\toks@,}%
247   \expandafter\KVS@@CommaComma\the\toks@\KVS@Nil
248 }

```

\KVS@@CommaComma

```

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

```

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

```

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

```

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

```

282 \def\KVS@SpaceEquals#1{%
283 \toks@\expandafter{\the\toks@#1}%
284 \expandafter\KVS@@SpaceEquals\the\toks@\KVS@Nil
285 }

```

`\KVS@@SpaceEquals`

```

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

```

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

```

294 \def\KVS@EqualsSpace{%
295 \toks@\expandafter{\the\toks@= }%
296 \expandafter\KVS@@EqualsSpace\the\toks@\KVS@Nil
297 }

```

`\KVS@@EqualsSpace`

```

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

```

### 3.6 Parsing key value lists

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

```
306 \def\kv@parse#1{%
307   \kv@normalize{#1}%
308   \expandafter\kv@parse@normalized\expandafter{\kv@list}%
309 }
```

`\kv@parse@normalized` #1: key value list  
#2: processor

```
310 \def\kv@parse@normalized#1#2{%
311   \KVS@Parse#1,\KVS@Nil{#2}%
312 }
```

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

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

`\KVS@Process` #1: key  
#2: value, =  
#3: processor

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

`\KVS@@Process` #1: key  
#2: value  
#3: processor

```
332 \def\KVS@@Process#1#2=\KVS@Nil#3{%
333   & \edef\kv@value{\etex@unexpanded{#2}}%
334   $ \begingroup
335   $ \toks@{#2}%
336   $ \xdef\KVS@Global{\the\toks@}%
337   $ \endgroup
338   $ \let\kv@value\KVS@Global
339   #3{#1}{#2}%
340 }
```

### 3.7 Parsing comma lists

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

```
341 \def\comma@parse#1{%
342   \comma@normalize{#1}%
343   \expandafter\comma@parse@normalized\expandafter{\comma@list}%
344 }
```

```

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

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

```

### 3.8 Processing key value pairs

```

\kv@processor@default
359 \def\kv@processor@default#1#2#3{%
360 & \unless\ifcsname KV@#1@#2\endcsname
361 $ \begingroup\expandafter\expandafter\expandafter\endgroup
362 $ \expandafter\ifx\csname KV@#1@#2\endcsname\relax
363 & \unless\ifcsname KVS@#1@handler\endcsname
364 $ \begingroup\expandafter\expandafter\expandafter\endgroup
365 $ \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
366   \kv@error@unknownkey{#1}{#2}%
367 \else
368   \csname KVS@#1@handler\endcsname{#2}{#3}%
369   \relax
370 \fi
371 \else
372   \ifx\kv@value\relax
373 & \unless\ifcsname KV@#1@#2@default\endcsname
374 $ \begingroup\expandafter\expandafter\expandafter\endgroup
375 $ \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
376   \kv@error@novalue{#1}{#2}%
377 \else
378   \csname KV@#1@#2@default\endcsname
379   \relax
380 \fi
381 \else
382   \csname KV@#1@#2\endcsname{#3}%
383 \fi
384 \fi
385 }

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

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

```

### 3.9 Error handling

```
\kv@error@novalue
392 \def\kv@error@novalue{%
393   \kv@error@generic{No value specified for}%
394 }

\kv@error@unknownkey
395 \def\kv@error@unknownkey{%
396   \kv@error@generic{Undefined}%
397 }

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

### 3.10 Do it all

```
\kvsetkeys
407 \def\kvsetkeys#1#2{%
408   \kv@parse{#2}{\kv@processor@default{#1}}%
409 }

410 \KVS@AtEnd
411 \</package>
```

## 4 Test

### 4.1 Catcode checks for loading

```
412 <*test1>
413 \catcode‘\{=1 %
414 \catcode‘\}=2 %
415 \catcode‘\#=6 %
416 \catcode‘\@=11 %
417 \expandafter\ifx\csname count@\endcsname\relax
418   \countdef\count@=255 %
419 \fi
420 \expandafter\ifx\csname @gobble\endcsname\relax
421   \long\def\@gobble#1{}%
422 \fi
423 \expandafter\ifx\csname @firstofone\endcsname\relax
424   \long\def\@firstofone#1{#1}%
425 \fi
426 \expandafter\ifx\csname loop\endcsname\relax
427   \expandafter\@firstofone
428 \else
429   \expandafter\@gobble
430 \fi
431 {%
432   \def\loop#1\repeat{%
433     \def\body{#1}%
434     \iterate
435   }%
```

```

436 \def\iterate{%
437   \body
438   \let\next\iterate
439   \else
440   \let\next\relax
441   \fi
442   \next
443 }%
444 \let\repeat=\fi
445 }%
446 \def\RestoreCatcodes{}
447 \count@=0 %
448 \loop
449   \edef\RestoreCatcodes{%
450     \RestoreCatcodes
451     \catcode\the\count@=\the\catcode\count@\relax
452   }%
453 \ifnum\count@<255 %
454   \advance\count@ 1 %
455 \repeat
456
457 \def\RangeCatcodeInvalid#1#2{%
458   \count@=#1\relax
459   \loop
460     \catcode\count@=15 %
461     \ifnum\count@<#2\relax
462       \advance\count@ 1 %
463     \repeat
464 }
465 \expandafter\ifx\csname LoadCommand\endcsname\relax
466 \def\LoadCommand{\input kvsetkeys.sty\relax}%
467 \fi
468 \def\Test{%
469   \RangeCatcodeInvalid{0}{47}%
470   \RangeCatcodeInvalid{58}{64}%
471   \RangeCatcodeInvalid{91}{96}%
472   \RangeCatcodeInvalid{123}{255}%
473   \catcode'\@=12 %
474   \catcode'\=0 %
475   \catcode'\{=1 %
476   \catcode'\}=2 %
477   \catcode'\#=6 %
478   \catcode'\[=12 %
479   \catcode'\]=12 %
480   \catcode'\%=14 %
481   \catcode'\ =10 %
482   \catcode\l3=5 %
483   \LoadCommand
484   \RestoreCatcodes
485 }
486 \Test
487 \csname @@end\endcsname
488 \end
489 </test1>

```

## 4.2 Macro tests

### 4.2.1 Preamble

```

490 <*test2>
491 \NeedsTeXFormat{LaTeX2e}
492 \nofiles
493 \documentclass{article}

```

```

494 \noetex\let\SavedUnexpanded\unexpanded
495 \noetex\let\unexpanded\UNDEFINED
496 \makeatletter
497 \chardef\KVS@TestMode=1 %
498 \makeatother
499 \usepackage{kvsetkeys}[2007/09/29]
500 \noetex\let\unexpanded\SavedUnexpanded
501 \usepackage{qstest}
502 \IncludeTests{*}
503 \LogTests{log}{*}{*}

```

#### 4.2.2 Time

```

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

```

#### 4.2.3 Test sets

```

541 \makeatletter
542 \def\@makeactive#1{%
543   \catcode'#1=13\relax
544 }
545 \@makeactive\,
546 \def,{\errmessage{COMMA}}
547 \@makeother\,
548 \@makeactive\=
549 \def={\errmessage{EQUALS}}
550 \@makeother\=
551
552 \begin{qstest}{normalize}{normalize,active-chars,space-removal}%

```

```

553 \def\Test#1#2{%
554   \@makeother\,%
555   \@makeother\=%
556   \scantokens{\toks@={#2}}%
557   \edef\Result{\the\toks@}%
558   \@makeother\,%
559   \@makeother\=%
560   \@Test{#1}%
561   \@makeactive\,%
562   \@Test{#1}%
563   \@makeactive\=%
564   \@Test{#1}%
565   \@makeother\,%
566   \@Test{#1}%
567   \@makeother\=%
568 }%
569 \def\@Test#1{%
570   \scantokens{\kv@normalize{#1}}%
571   \expandafter\expandafter\expandafter\Expect
572   \expandafter\expandafter\expandafter
573   {\expandafter\kv@list\expandafter}\expandafter{\Result}%
574   \Expect*{\ifx\kv@list\Result true\else false\fi}{true}%
575 }%
576 \Test{}{,}%
577 \Test{,}{,}%
578 \Test{,,}{,}%
579 \Test{,,,}{,}%
580 \Test{ , }{,}%
581 \Test{{a}}{,{a},}%
582 \Test{,{a}}{,{a},}%
583 \Test{{a},}{,{a},}%
584 \Test{{a},{b}}{,{a},{b},}%
585 \Test{{b}={c},{}=,{d}={c},{b}={c},{}=,{d}={c},{d}={c},}%
586 \Test{{}}{,{},}%
587 \Test{{},{},{}}{,{},{},}%
588 \Test{=}{,=,}%
589 \Test{=,=}{,=,=,}%
590 \def\TestSet#1{%
591   \Test{#1#1}{,}%
592   \Test{#1#1,#1#1}{,}%
593   \Test{#1#1,#1#1,#1#1}{,}%
594   \Test{#1#1#1#1#1}{,}%
595   \Test{{a}#1#1=#1#1{b}}{,{a}={b},}%
596 }%
597 \TestSet{ }%
598 \begingroup
599   \let\saved@normalize\kv@normalize
600   \def\kv@normalize#1{%
601     \saved@normalize{#1}%
602     \@onelevel@sanitize\kv@list
603     \@onelevel@sanitize\Result
604   }%
605   \Test{#,#=#,{#}={#},{#}={#},{#}={#},{#}={#},{#}={#},{#}={#},}%
606 \endgroup
607 \begingroup
608   \def\Test#1#2{%
609     \edef\Result{#2}%
610     \@Test{#1}%
611   }%
612   \Test{{ a = b }}{,{ a = b },}%
613   \@makeactive\,%
614   \Test{{,}}{\string,{\noexpand,}\string,}%

```



```

615 \makeother\,%
616 \makeactive\=%
617 \Test{a={}}{,a\string={\noexpand=},}%
618 \endgroup
619 \Test{a=b}{,a=b,}%
620 \Test{a={b}}{,a={b},}%
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}{,a,}%
626 \Test{ a}{,a,}%
627 \Test{a }{,a,}%
628 \Test{ a }{,a,}%
629 \Test{, a }{,a,}%
630 \Test{, a b }{,a b,}%
631 \Test{,a }{,a,}%
632 \Test{ a =}{,a=,}%
633 \Test{ a = }{,a=,}%
634 \Test{a =}{,a=,}%
635 \Test{{a} =}{,{a}=,}%
636 \Test{{a}= {} }{,{a}={},}%
637 \Test{, a = {} }{,a={},}%
638 \Test{a,,b}{,a,b,}%
639 \Test{a=\fi}{,a=\fi,}%
640 \Test{a=\iffalse}{,a=\iffalse,}%
641 \Test{a=\iffalse,b=\fi}{,a=\iffalse,b=\fi,}%
642 \end{qstest}
643
644 \begin{qstest}{parse}{parse,brace-removal}
645 \def\Processor#1#2{%
646 \expandafter\Expect\expandafter{\kv@key}{#1}%
647 \toks@{#2}%
648 \edef\x{\the\toks@}%
649 \ifx\kv@value\relax
650 \Expect*{\the\toks@}{}%
651 \def\Value{<>}%
652 \else
653 \edef\Value{[\the\toks@]}%
654 \@onelevel@sanitize\Value
655 \fi
656 \toks@{#1}%
657 \ifx\Result\@empty
658 \edef\Result{[\the\toks@]=\Value}%
659 \else
660 \edef\Result{\Result,[\the\toks@]=\Value}%
661 \fi
662 \@onelevel@sanitize\Result
663 }%
664 \def\Test#1#2{%
665 \let\Result\@empty
666 \kv@parse{#1}\Processor
667 \Expect*{\Result}{#2}%
668 }%
669 \Test{}{}%
670 \Test{{}}{}%
671 \Test{{{}}}{[]=<>}%
672 \Test{{{}}}{[{}]=<>}%
673 \Test{a}{[a]=<>}%
674 \Test{{a}}{[a]=<>}%
675 \Test{{a}}{[a]=<>}%
676 \Test{{{a}}}{[a]=<>}%

```

```

677 \Test{{{a}}}{[a]=<>}%
678 \Test{a=}{[a]=[]}%
679 \Test{{a}=}{[a]=[]}%
680 \Test{{{a}}=}{[a]=[]}%
681 \Test{a={}}{[a]=[]}%
682 \Test{{a}={}}{[a]={}}%
683 \Test{a=b}{[a]=[b]}%
684 \Test{a=fi}{[a]=[fi]}%
685 \Test{a=iffalse}{[a]=[iffalse]}%
686 \Test{a=iffalse,b=fi}{[a]=[iffalse],[b]=[fi]}%
687 \Test{{ a = b }}{[ a ]=[ b ]}%
688 \Test{{{ a = b }}}{[ a = b ]=<>}%
689 \end{qstest}
690
691 \begin{qstest}{comma}{comma,parse}
692 \def\Processor#1{%
693 \expandafter\Expect\expandafter{\comma@entry}{#1}%
694 \toks@{#1}%
695 \ifx\Result\@empty
696 \edef\Result{[\the\toks@]}%
697 \else
698 \edef\Result{\Result,[\the\toks@]}%
699 \fi
700 \@onelevel@sanitize\Result
701 }%
702 \def\Test#1#2{%
703 \let\Result\@empty
704 \comma@parse{#1}\Processor
705 \Expect*\Result}{#2}%
706 }%
707 \tracingmacros=1
708 \Test{}{}%
709 \Test{{}}{}%
710 \Test{{{}}}{{{}}}%
711 \Test{a}{[a]}%
712 \Test{{a}}{[a]}%
713 \Test{{{a}}}{[a]}%
714 \Test{a=}{[a=]}%
715 \Test{a=fi}{[a=fi]}%
716 \Test{a=iffalse}{[a=iffalse]}%
717 \Test{\iffalse,fi}{[\iffalse],[fi]}%
718 \Test{ a , b , c }{[a],[b],[c]}%
719 \Test{ { } , { } , { } , { } }{[ ],[ ],[ ],[ ]}%
720 \Test{ {{ } },{{ } }, {{ } }, {{ } } , {{ } } }{[{}],[{}],[{}],[{}],[{}]}%
721 \end{qstest}
722
723 \begin{document}
724 \end{document}
725 </test2>

```

## 5 Installation

### 5.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

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

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

---

<sup>1</sup><http://ftp.ctan.org/tex-archive/>

**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](#)

*TDS* refers to the standard “A Directory Structure for T<sub>E</sub>X 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<sub>E</sub>X:

```
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>test/kvsetkeys-test1.tex</code>	→ <code>doc/latex/oberdiek/test/kvsetkeys-test1.tex</code>
<code>test/kvsetkeys-test2.tex</code>	→ <code>doc/latex/oberdiek/test/kvsetkeys-test2.tex</code>
<code>test/kvsetkeys-test3.tex</code>	→ <code>doc/latex/oberdiek/test/kvsetkeys-test3.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`.

## 5.4 Refresh file name databases

If your T<sub>E</sub>X distribution (teT<sub>E</sub>X, mikT<sub>E</sub>X, ...) relies on file name databases, you must refresh these. For example, teT<sub>E</sub>X 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 L<sup>A</sup>T<sub>E</sub>X.** The .dtx chooses its action depending on the format:

**plain-T<sub>E</sub>X:** Run docstrip and extract the files.

**L<sup>A</sup>T<sub>E</sub>X:** Generate the documentation.

If you insist on using L<sup>A</sup>T<sub>E</sub>X for docstrip (really, docstrip does not need L<sup>A</sup>T<sub>E</sub>X), 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<sup>A</sup>T<sub>E</sub>X:

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

## 6 References

- [1] 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.
- $\varepsilon$ -T<sub>E</sub>X is used if available.
- Tests added for normalizing and parsing.

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.

21

\KVS@@CommaComma	247, 249		
\KVS@@CommaSpace	235, 237		
\KVS@@Equals	262, 264, 278		
\KVS@@EqualsSpace	296, 298		
\KVS@@Process	329, 332		
\KVS@@SpaceComma	223, 225		
\KVS@@SpaceEquals	284, 286		
\KVS@AtEnd	125, 126, 410		
\KVS@Comma	183, 197, 205		
\KVS@CommaComma	186, 200, 245		
\KVS@CommaParse	346, 348		
\KVS@CommaSpace	185, 199, 233		
\KVS@Empty	165, 174, 266		
\KVS@Equals	187, 256		
\KVS@EqualsSpace	189, 294		
\KVS@FirstOfTwo	166, 175, 267		
\KVS@Global	190, 192, 201, 203, 336, 338		
\KVS@IfEmpty	168, 215, 226, 238, 251, 276, 287, 299, 314, 318, 325, 349, 354		
\KVS@Nil	211, 213, 217, 223, 225, 230, 235, 237, 242, 247, 249, 253, 262, 264, 278, 284, 286, 291, 296, 298, 303, 311, 313, 316, 320, 323, 329, 332, 346, 348, 356		
\KVS@Parse	311, 313		
\KVS@Process	316, 323		
\KVS@SecondOfTwo	167, 177, 269		
\KVS@SetFamilyHandler	387, 389		
\KVS@SpaceComma	184, 198, 221		
\KVS@SpaceEquals	188, 282		
\KVS@Temp	169, 172, 174, 265, 266		
\KVS@TestMode	497		
\kvsetkeys	4, 14, 407		
<b>L</b>			
\lccode	206, 207, 257, 258		
\LoadCommand	466, 483		
\LogTests	503		
\loop	432, 448, 459		
\lowercase	208, 259		
<b>M</b>			
\makeatletter	7, 496, 507, 541		
\makeatother	32, 498, 539		
\mbox	36		
\MessageBreak	402, 403		
<b>N</b>			
\NeedsTeXFormat	491		
\newcommand	8, 511, 516, 520, 521		
\newcount	508, 509		
\next	438, 440, 442		
\nofiles	492		
\number	513		
<b>P</b>			
\PackageInfo	73		
\pdfelapsedtime	522		
\pdfresettimer	518		
\PrintTime	511, 524, 537		
\Processor	645, 666, 692, 704		
\ProvidesPackage	106		
<b>Q</b>			
\qqquad	36		
\qstest	526, 528		
<b>R</b>			
\RangeCatcodeInvalid	457, 469, 470, 471, 472		
\renewcommand	517		
\repeat	432, 444, 455, 463		
\RequirePackage	146, 147		
\RestoreCatcodes	446, 449, 450, 484		
\Result	557, 573, 574, 603, 609, 657, 658, 660, 662, 665, 667, 695, 696, 698, 700, 703, 705		
<b>S</b>			
\saved@endqstest	527, 534		
\saved@normalize	599, 601		
\saved@qstest	526, 529		
\SavedUnexpanded	494, 500		
\scantokens	556, 570		
\space	25, 513		
\StartTime	516, 530		
\StopTime	521, 533		
\strip@pt	513		
\SummaryTime	508, 510, 523, 537		
<b>T</b>			
\tag	8, 37, 39, 40, 42, 44		
\Test	468, 486, 553, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 591, 592, 593, 594, 595, 605, 608, 612, 614, 617, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 664, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 702, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720		
\TestSet	590, 597		
\TestTime	509, 522, 523, 524		
\textgreater	16		
\textless	16		
\texttt	15		
\the	16, 24, 113, 114, 115, 116, 127, 172, 190, 201, 211, 214, 222, 223, 230, 234, 235, 242, 246, 247, 253, 262, 265, 274, 283, 284, 291, 295, 296, 303, 336, 451, 557, 648, 650, 653, 658, 660, 696, 698		
\TimeDescription	517, 520, 524		
\TMP@EnsureCode	124, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140		
\toks@	12, 16, 23, 24, 171, 172, 182, 190, 196, 201, 210, 211, 214, 222, 223, 227, 229, 230, 234, 235, 239, 241, 242, 246, 247, 250, 253, 261, 262, 265, 272,		

274, 283, 284, 288, 290, 291, 295, 296, 300, 302, 303, 335, 336, 556, 557, 647, 648, 650, 653, 656, 658, 660, 694, 696, 698		<b>V</b>	
	<code>\Value</code> .....	651, 653, 654, 658, 660	
		<b>W</b>	
<code>\tracingmacros</code> .....	707	<code>\write</code> .....	70, 94
<code>\typeout</code> .....	512		
		<b>X</b>	
<b>U</b>		<code>\x</code> .....	55, 57, 59, 69, 73, 75, 93, 98, 105, 111, 119, 648
<code>\UNDEFINED</code> .....	495		
<code>\unexpanded</code> .....	494, 495, 500		
<code>\unless</code> .....	360, 363, 373	<b>Z</b>	
<code>\usepackage</code> .....	3, 4, 5, 499, 501	<code>\z@</code> .....	510