

The kvsetkeys package

Heiko Oberdiek*

2019/12/15 v1.18

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	3
1.3	Parsing key value lists	3
1.4	Processing key value pairs	4
1.4.1	Processing similar to keyval	4
1.4.2	Processing similar to <code>\setkeys*</code> of package xkeyval	5
1.5	Default family handler	5
1.6	Put it all together	5
1.7	Comma separated lists	6
2	Example	7
3	Implementation	8
3.1	Identification	8
3.2	Package loading	10
3.3	Check for ϵ -TeX	10
3.4	Generic help macros	10
3.5	Normalizing	11
3.6	Parsing key value lists	13
3.7	Parsing comma lists	15
3.8	Processing key value pairs	15
3.9	Error handling	18
3.10	Do it all	18
4	Installation	19
4.1	Download	19
4.2	Bundle installation	20
4.3	Package installation	20
4.4	Refresh file name databases	20
4.5	Some details for the interested	20

*Please report any issues at <https://github.com/ho-tex/kvsetkeys/issues>

5	References	21
6	History	21
	[2006/03/06 v1.0]	21
	[2006/10/19 v1.1]	21
	[2007/09/09 v1.2]	21
	[2007/09/29 v1.3]	21
	[2009/07/19 v1.4]	21
	[2009/07/30 v1.5]	22
	[2009/12/12 v1.6]	22
	[2009/12/22 v1.7]	22
	[2010/01/28 v1.8]	22
	[2010/03/01 v1.9]	22
	[2011/01/30 v1.10]	22
	[2011/03/03 v1.11]	22
	[2011/04/05 v1.12]	22
	[2011/04/07 v1.13]	22
	[2011/06/15 v1.14]	22
	[2011/10/18 v1.15]	22
	[2012/04/25 v1.16]	23
	[2016/05/16 v1.17]	23
	[2019/12/15 v1.18]	23
7	Index	23

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}}} → value
\setkeys{fam}{key={{value}}} → {value}
\setkeys{fam}{key= {{value}}} → {{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}} → value
\kvsetkeys{fam}{key={{value}}} → {value}
\kvsetkeys{fam}{key= {{value}}} → {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` $\{\langle key\ value\ list\rangle\}$ $\{\langle processor\rangle\}$

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 $\langle processor\rangle$:

$\langle processor\rangle$ $\{\langle key\rangle\}$ $\{\langle value\rangle\}$

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 ( $\langle key\rangle$ ,  $\langle value\rangle$ ) in ( $\langle key\ value\ list\rangle$ )
  \kv@key :=  $\langle key\rangle$ 
  \kv@value :=  $\langle value\rangle$ 
   $\langle processor\rangle$   $\{\langle key\rangle\}$   $\{\langle value\rangle\}$ 
```

`\kv@break`

Since version 2011/03/03 v1.11 `\kv@break` can be called inside the $\langle processor\rangle$ of `\kv@parse` or `\kv@parse@normalized`, then the processing is stopped and the following entries discarded.

1.4 Processing key value pairs

Key value pairs can be processed in many different ways. For example, the processor for `\kvsetkeys` works similar to `\setkeys` of package `keyval`. There unknown keys raise an error.

Package `xkeyval` also knows a star form of `\setkeys` that stores unknown keys in an internal macro for further processing with `\setrmkeys` and similar macros. This feature is covered by processor `\kv@processor@known`.

1.4.1 Processing similar to `keyval`

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

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. Both $\langle family\rangle$ and $\langle key\rangle$ may contain package `babel`'s shorthands (since 2011/04/07 v1.13).

Since 2011/10/18 v1.15 the family handler can reject the successful handling of a key by calling `\kv@handled@false`.

Since 2019/12/15 v1.18 `\kv@processor@default` also defines macro `\kv@fam` with meaning $\langle family\rangle$ for convenience.

1.4.2 Processing similar to `\setkeys*` of package `xkeyval`

`\kv@processor@known {<family>} {<cmd>} {<key>} {<value>}`

The key value processor `\kv@processor@known` behaves similar to `\kv@processor@default`. If the `<key>` exists in the `<family>` its code is called, otherwise the family handler is tried. If the family handler is not set or cannot handle the key, the unknown key value pair is added to the macro `<cmd>`. Since 2011/10/18 v1.15.

The behaviour in pseudo code:

```
if <key> exists
  call the keyval code of <key>
else
  if <handler> for <family> exists
    handled = true
    <handler> {<key>} {<value>}
    if handled
      else
        add "{<key>}={<value>}" to {<cmd>}
      fi
    else
      add "{<key>}={<value>}" to {<cmd>}
      raise unknown key error
    fi
  fi
```

Since 2019/12/15 v1.18 `\kv@processor@known` also defines macro `\kv@fam` with meaning `<family>` for convenience.

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`.

`\kv@unset@family@handler {<family>}`

It removes the family handler for `<family>`. Since 2011/10/18 v1.15.

1.6 Put it all together

`\kvsetkeys {<family>} {<key value list>}`

Macro `\kvsetkeys` processes the `<key value list>` with the standard processor `\kv@processor@default`:

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

```
\kvsetknownkeys {⟨family⟩} {⟨cmd⟩} {⟨key value list⟩}
```

Macro `\kvsetknownkeys` processes the `⟨key value list⟩` with processor `\kv@processor@known`. All key value pairs with keys that are not known in `⟨family⟩` are stored in macro `⟨cmd⟩`. A previous contents of macro `⟨cmd⟩` will be overwritten. If all keys can be handled, `⟨cmd⟩` will be empty, otherwise it contains a key value list of unhandled key value pairs. Since 2011/10/18 v1.15.

Pseudo code:

```
create macro ⟨cmdaux⟩ with unique name (inside the current group)
\def⟨cmdaux⟩{}
\kv@parse {⟨key value list⟩}{\kv@processor@known {⟨family⟩} {⟨cmdaux⟩}}
\let⟨cmd⟩=⟨cmdaux⟩
```

```
\kvsetkeys@expandafter {⟨family⟩} {⟨list cmd⟩}
\kvsetknownkeys@expandafter {⟨family⟩} {⟨cmd⟩} {⟨list cmd⟩}
```

Both macros behave like the counterparts without suffix `@expandafter`. The difference is that the key value list is given as macro that is expanded once. Since 2011/10/18 v1.15.

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`.

Since version 2011/03/03 v1.11 `\comma@break` can be called inside the $\langle processor \rangle$ of `\comma@parse` or `\comma@parse@normalized`, then the processing is stopped and the following entries discarded.

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  $\langle *example \rangle$ 
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^AT_EX.

```
49 \begingroup\catcode61\catcode48\catcode32=10\relax%
50 \catcode13=5 % ^M
51 \endlinechar=13 %
52 \catcode35=6 % #
53 \catcode39=12 % '
54 \catcode44=12 % ,
55 \catcode45=12 % -
56 \catcode46=12 % .
57 \catcode58=12 % :
58 \catcode64=11 % @
59 \catcode123=1 % {
60 \catcode125=2 % }
61 \expandafter\let\expandafter\x\csname ver@kvsetkeys.sty\endcsname
62 \ifx\x\relax % plain-TeX, first loading
63 \else
64 \def\empty{}%
65 \ifx\x\empty % LaTeX, first loading,
66 % variable is initialized, but \ProvidesPackage not yet seen
67 \else
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 \aftergroup\endinput
77 \fi
78 \fi
79 \endgroup%
```

Package identification:

```
80 \begingroup\catcode61\catcode48\catcode32=10\relax%
81 \catcode13=5 % ^M
82 \endlinechar=13 %
83 \catcode35=6 % #
84 \catcode39=12 % '
85 \catcode40=12 % (
86 \catcode41=12 % )
87 \catcode44=12 % ,
88 \catcode45=12 % -
89 \catcode46=12 % .
90 \catcode47=12 % /
91 \catcode58=12 % :
92 \catcode64=11 % @
93 \catcode91=12 % [
94 \catcode93=12 % ]
95 \catcode123=1 % {
96 \catcode125=2 % }
```

```

97 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
98 \def\x#1#2#3[#4]{\endgroup
99 \immediate\write-1{Package: #3 #4}%
100 \xdef#1{#4}%
101 }%
102 \else
103 \def\x#1#2[#3]{\endgroup
104 #2[#{#3}]%
105 \ifx#1\undefined
106 \xdef#1{#3}%
107 \fi
108 \ifx#1\relax
109 \xdef#1{#3}%
110 \fi
111 }%
112 \fi
113 \expandafter\x\csname ver@kvsetkeys.sty\endcsname
114 \ProvidesPackage[kvsetkeys]%
115 [2019/12/15 v1.18 Key value parser (HO)]%
116 \begingroup\catcode61\catcode48\catcode32=10\relax%
117 \catcode13=5 % ^M
118 \endlinechar=13 %
119 \catcode123=1 % {
120 \catcode125=2 % }
121 \catcode64=11 % @
122 \def\x{\endgroup
123 \expandafter\edef\csname KVS@AtEnd\endcsname{%
124 \endlinechar=\the\endlinechar\relax
125 \catcode13=\the\catcode13\relax
126 \catcode32=\the\catcode32\relax
127 \catcode35=\the\catcode35\relax
128 \catcode61=\the\catcode61\relax
129 \catcode64=\the\catcode64\relax
130 \catcode123=\the\catcode123\relax
131 \catcode125=\the\catcode125\relax
132 }%
133 }%
134 \x\catcode61\catcode48\catcode32=10\relax%
135 \catcode13=5 % ^M
136 \endlinechar=13 %
137 \catcode35=6 % #
138 \catcode64=11 % @
139 \catcode123=1 % {
140 \catcode125=2 % }
141 \def\TMP@EnsureCode#1#2{%
142 \edef\KVS@AtEnd{%
143 \KVS@AtEnd
144 \catcode#1=\the\catcode#1\relax
145 }%
146 \catcode#1=#2\relax
147 }
148 \TMP@EnsureCode{36}{3}% $
149 \TMP@EnsureCode{38}{4}% &
150 \TMP@EnsureCode{39}{12}% '
151 \TMP@EnsureCode{43}{12}% +
152 \TMP@EnsureCode{44}{12}% ,
153 \TMP@EnsureCode{45}{12}% -
154 \TMP@EnsureCode{46}{12}% .

```

```

155 \TMP@EnsureCode{47}{12}% /
156 \TMP@EnsureCode{91}{12}% [
157 \TMP@EnsureCode{93}{12}% ]
158 \TMP@EnsureCode{94}{7}% ^ (superscript)
159 \TMP@EnsureCode{96}{12}% ‘
160 \TMP@EnsureCode{126}{13}% ~ (active)
161 \edef\KVS@AtEnd{\KVS@AtEnd\noexpand\endinput}

```

3.2 Package loading

```

162 \begingroup\expandafter\expandafter\expandafter\endgroup
163 \expandafter\ifx\csname RequirePackage\endcsname\relax
164   \def\TMP@RequirePackage#1[#2]{%
165     \begingroup\expandafter\expandafter\expandafter\endgroup
166     \expandafter\ifx\csname ver@#1.sty\endcsname\relax
167       \input #1.sty\relax
168     \fi
169   }%
170   \TMP@RequirePackage{infwarerr}[2007/09/09]%
171   \let\PackageError\@PackageError
172 \else
173 \fi

174 \expandafter\ifx\csname toks@\endcsname\relax
175   \toksdef\toks@=0 %
176 \fi

```

3.3 Check for ϵ -TeX

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

```

177 \ifx\numexpr\@undefined
178   \catcode'\$=9 % ignore
179   \catcode'\&=14 % comment
180 \else % e-TeX
181   \catcode'\$=14 % comment
182   \catcode'\&=9 % ignore
183 \fi

```

3.4 Generic help macros

```

\KVS@Empty
184 \def\KVS@Empty{}

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

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

\KVS@IfEmpty
187 \long\def\KVS@IfEmpty#1{%
188 & \edef\KVS@Temp{\unexpanded{#1}}%
189 $ \begingroup
190 $   \toks@{#1}%
191 $   \edef\KVS@Temp{\the\toks@}%
192 $ \expandafter\endgroup
193 \ifx\KVS@Temp\KVS@Empty
194   \expandafter\KVS@FirstOfTwo
195 \else

```

```

196 \expandafter\KVS@SecondOfTwo
197 \fi
198 }

```

3.5 Normalizing

`\kv@normalize`

```

199 \long\def\kv@normalize#1{%
200 \begingroup
201 \toks@{,#1,}%
202 \KVS@Comma
203 \KVS@SpaceComma
204 \KVS@CommaSpace
205 \KVS@CommaComma
206 \KVS@Equals
207 \KVS@SpaceEquals
208 \KVS@EqualsSpace
209 \xdef\KVS@Global{\the\toks@}%
210 \endgroup
211 \let\kv@list\KVS@Global
212 }

```

`\comma@normalize`

```

213 \def\comma@normalize#1{%
214 \begingroup
215 \toks@{,#1,}%
216 \KVS@Comma
217 \KVS@SpaceComma
218 \KVS@CommaSpace
219 \KVS@CommaComma
220 \xdef\KVS@Global{\the\toks@}%
221 \endgroup
222 \let\comma@list\KVS@Global
223 }

```

`\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.

```

224 \begingroup
225 \lccode'\,=\,%
226 \lccode'\~=\,%
227 \lowercase{\endgroup
228 \def\KVS@Comma{%
229 \toks@\expandafter{\expandafter}\expandafter
230 \KVS@@Comma\the\toks@\~\KVS@Nil
231 }%
232 \long\def\KVS@@Comma#1~#2\KVS@Nil{%
233 \toks@\expandafter{\the\toks@#1}%
234 \KVS@IfEmpty{#2}{-%
235 }-%
236 \KVS@@Comma,#2\KVS@Nil
237 }%
238 }%
239 }

```

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

```

240 \def\KVS@SpaceComma#1{%
241 \def\KVS@SpaceComma{%

```

```

242   \expandafter\KVS@@SpaceComma\the\toks@#1,\KVS@Nil
243   }%
244 }
245 \KVS@@SpaceComma{ }

\KVS@@SpaceComma
246 \long\def\KVS@@SpaceComma#1 ,#2\KVS@Nil{%
247   \KVS@IfEmpty{#2}{%
248     \toks@{#1}%
249   }{%
250     \KVS@@SpaceComma#1,#2\KVS@Nil
251   }%
252 }

\KVS@CommaSpace Removes spaces after the comma, may add commas at the end.
253 \def\KVS@CommaSpace{%
254   \expandafter\KVS@@CommaSpace\the\toks@, \KVS@Nil
255 }

\KVS@@CommaSpace
256 \long\def\KVS@@CommaSpace#1 ,#2\KVS@Nil{%
257   \KVS@IfEmpty{#2}{%
258     \toks@{#1}%
259   }{%
260     \KVS@@CommaSpace#1,#2\KVS@Nil
261   }%
262 }

\KVS@CommaComma Replaces multiple commas by one comma.
263 \def\KVS@CommaComma{%
264   \expandafter\KVS@@CommaComma\the\toks@,\KVS@Nil
265 }

\KVS@@CommaComma
266 \long\def\KVS@@CommaComma#1 , ,#2\KVS@Nil{%
267   \KVS@IfEmpty{#2}{%
268     \toks@{#1,}% (!)
269   }{%
270     \KVS@@CommaComma#1,#2\KVS@Nil
271   }%
272 }

\KVS@Equals Converts active equals signs into catcode other characters.
273 \begingroup
274   \lccode'\=='\=%
275   \lccode'\~='\=%
276 \lowercase{\endgroup
277   \def\KVS@Equals{%
278     \toks@\expandafter{\expandafter}\expandafter
279     \KVS@@Equals\the\toks@~\KVS@Nil
280   }%
281   \long\def\KVS@@Equals#1~#2\KVS@Nil{%
282     \edef\KVS@Temp{\the\toks@}%
283     \ifx\KVS@Temp\KVS@Empty
284       \expandafter\KVS@FirstOfTwo
285     \else
286       \expandafter\KVS@SecondOfTwo

```

```

287 \fi
288 {%
289 \toks@{#1}%
290 }{%
291 \toks@\expandafter{\the\toks@=#1}%
292 }%
293 \KVS@IfEmpty{#2}{%
294 }{%
295 \KVS@@Equals#2\KVS@Nil
296 }%
297 }%
298 }

```

\KVS@SpaceEquals Removes spaces before the equals sign.

```

299 \def\KVS@SpaceEquals#1{%
300 \def\KVS@SpaceEquals{%
301 \expandafter\KVS@@SpaceEquals\the\toks@#1=\KVS@Nil
302 }%
303 }
304 \KVS@SpaceEquals{ }

```

\KVS@@SpaceEquals

```

305 \long\def\KVS@@SpaceEquals#1 =#2\KVS@Nil{%
306 \KVS@IfEmpty{#2}{%
307 \toks@{#1}%
308 }{%
309 \KVS@@SpaceEquals#1=#2\KVS@Nil
310 }%
311 }

```

\KVS@EqualsSpace Removes spaces after the equals sign.

```

312 \def\KVS@EqualsSpace{%
313 \expandafter\KVS@@EqualsSpace\the\toks@= \KVS@Nil
314 }

```

\KVS@@EqualsSpace

```

315 \long\def\KVS@@EqualsSpace#1= #2\KVS@Nil{%
316 \KVS@IfEmpty{#2}{%
317 \toks@{#1}%
318 }{%
319 \KVS@@EqualsSpace#1=#2\KVS@Nil
320 }%
321 }

```

3.6 Parsing key value lists

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

```

322 \long\def\kv@parse#1{%
323 \kv@normalize{#1}%
324 \expandafter\kv@parse@normalized\expandafter{\kv@list}%
325 }

```

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

#2: processor

```

326 \long\def\kv@parse@normalized#1#2{%
327 \KVS@Parse#1,\KVS@Nil{#2}%
328 }

```

```

\KVS@Parse #1,#2: key value list
#3: processor
329 \long\def\KVS@Parse#1,#2\KVS@Nil#3{%
330 \KVS@IfEmpty{#1}{%
331 }{%
332 \KVS@Process#1=\KVS@Nil{#3}%
333 }%
334 \KVS@MaybeBreak
335 \KVS@IfEmpty{#2}{%
336 }{%
337 \KVS@Parse#2\KVS@Nil{#3}%
338 }%
339 }

\KVS@Process #1: key
#2: value, =
#3: processor
340 \long\def\KVS@Process#1=#2\KVS@Nil#3{%
341 \let\KVS@MaybeBreak\relax
342 \def\kv@key{#1}%
343 \KVS@IfEmpty{#2}{%
344 \let\kv@value\relax
345 #3{#1}{}%
346 }{%
347 \KVS@@Process{#1}#2\KVS@Nil{#3}%
348 }%
349 }

\KVS@@Process #1: key
#2: value
#3: processor
350 \long\def\KVS@@Process#1#2=\KVS@Nil#3{%
351 & \edef\kv@value{\unexpanded{#2}}%
352 $ \begingroup
353 $ \toks@{#2}%
354 $ \xdef\KVS@Global{\the\toks@}%
355 $ \endgroup
356 $ \let\kv@value\KVS@Global
357 #3{#1}{#2}%
358 }

\KVS@MaybeBreak
359 \let\KVS@MaybeBreak\relax

\KVS@break
360 \def\KVS@break#1#2#3#4{%
361 \let\KVS@MaybeBreak\relax
362 }

\kv@break
363 \def\kv@break{%
364 \let\KVS@MaybeBreak\KVS@break
365 }

```

3.7 Parsing comma lists

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

```
366 \def\comma@parse#1{%
367   \comma@normalize{#1}%
368   \expandafter\comma@parse@normalized\expandafter{\comma@list}%
369 }
```

`\comma@parse@normalized` #1: comma list
#2: processor

```
370 \def\comma@parse@normalized#1#2{%
371   \KVS@CommaParse#1,\KVS@Nil{#2}%
372 }
```

`\KVS@CommaParse` #1,#2: comma list
#3: processor

```
373 \def\KVS@CommaParse#1,#2\KVS@Nil#3{%
374   \KVS@IfEmpty{#1}{%
375     }{%
376     \def\comma@entry{#1}%
377     #3{#1}%
378     }%
379   \KVS@MaybeBreak
380   \KVS@IfEmpty{#2}{%
381     }{%
382     \KVS@CommaParse#2\KVS@Nil{#3}%
383     }%
384 }
```

`\comma@break`

```
385 \def\comma@break{%
386   \let\KVS@MaybeBreak\KVS@break
387 }
```

3.8 Processing key value pairs

`\kv@handled@false` The handler can call `\kv@handled@false` or `\kv@handled@true` so report failure or success. The default is success (compatibility for versions before 2011/10/18 v1.15).

```
388 \def\kv@handled@false{%
389   \let\ifkv@handled@iffalse
390 }
```

`\kv@handled@true`

```
391 \def\kv@handled@true{%
392   \let\ifkv@handled@iftrue
393 }
```

`\ifkv@handled@`

```
394 \kv@handled@true
```

`\kv@processor@default`

```
395 \def\kv@processor@default#1#2{%
396   \begingroup
397   \csname @safe@activetrue\endcsname
398   \let\ifincsname\iftrue
399   \edef\KVS@temp{\endgroup
```

```

400     \noexpand\KVS@ProcessorDefault{#1}{#2}%
401   }%
402   \KVS@temp
403 }

```

\KVS@ProcessorDefault

```

404 \long\def\KVS@ProcessorDefault#1#2#3{%
405   \def\kv@fam{#1}%
406 & \unless\ifcsname KV@#1@#2\endcsname
407 $ \begingroup\expandafter\expandafter\expandafter\endgroup
408 $ \expandafter\ifx\csname KV@#1@#2\endcsname\relax
409 & \unless\ifcsname KVS@#1@handler\endcsname
410 $ \begingroup\expandafter\expandafter\expandafter\endgroup
411 $ \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
412   \kv@error@unknownkey{#1}{#2}%
413   \else
414     \kv@handled@true
415     \csname KVS@#1@handler\endcsname{#2}{#3}%
416     \relax
417     \ifkv@handled@
418     \else
419       \kv@error@unknownkey{#1}{#2}%
420     \fi
421   \fi
422 \else
423   \ifx\kv@value\relax
424 & \unless\ifcsname KV@#1@#2@default\endcsname
425 $ \begingroup\expandafter\expandafter\expandafter\endgroup
426 $ \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
427   \kv@error@novalue{#1}{#2}%
428   \else
429     \csname KV@#1@#2@default\endcsname
430     \relax
431   \fi
432 \else
433   \csname KV@#1@#2\endcsname{#3}%
434 \fi
435 \fi
436 }

```

\kv@processor@known

```

437 \def\kv@processor@known#1#2#3{%
438   \begingroup
439     \csname @safe@activetrue\endcsname
440     \let\ifincsname\iftrue
441     \edef\KVS@temp{\endgroup
442       \noexpand\KVS@ProcessorKnown{#1}\noexpand#2{#3}%
443     }%
444   \KVS@temp
445 }

```

\KVS@ProcessorKnown

```

446 \long\def\KVS@ProcessorKnown#1#2#3#4{%
447   \def\kv@fam{#1}%
448 & \unless\ifcsname KV@#1@#3\endcsname
449 $ \begingroup\expandafter\expandafter\expandafter\endgroup
450 $ \expandafter\ifx\csname KV@#1@#3\endcsname\relax
451 & \unless\ifcsname KVS@#1@handler\endcsname

```

```

452 $ \begingroup\expandafter\expandafter\expandafter\endgroup
453 $ \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
454 \KVS@AddUnhandled#2{#3}{#4}%
455 \else
456 \kv@handled@true
457 \csname KVS@#1@handler\endcsname{#3}{#4}%
458 \relax
459 \ifkv@handled@
460 \else
461 \KVS@AddUnhandled#2{#3}{#4}%
462 \fi
463 \fi
464 \else
465 \ifx\kv@value\relax
466 & \unless\ifcsname KV@#1@#2@default\endcsname
467 $ \begingroup\expandafter\expandafter\expandafter\endgroup
468 $ \expandafter\ifx\csname KV@#1@#3@default\endcsname\relax
469 \kv@error@novalue{#1}{#3}%
470 \else
471 \csname KV@#1@#3@default\endcsname
472 \relax
473 \fi
474 \else
475 \csname KV@#1@#3\endcsname{#4}%
476 \fi
477 \fi
478 }

```

\KVS@AddUnhandled

```

479 \long\def\KVS@AddUnhandled#1#2#3{%
480 & \edef#1{%
481 & \ifx#1\KVS@empty
482 & \else
483 & \unexpanded{#1},%
484 & \fi
485 & \unexpanded{{#2}={#3}}%
486 & }%
487 $ \begingroup
488 $ \ifx#1\KVS@empty
489 $ \toks@{#2}={#3}%
490 $ \else
491 $ \toks@\expandafter{#1,#2}={#3}%
492 $ \fi
493 $ \xdef\KVS@Global{\the\toks@}%
494 $ \endgroup
495 $ \let#1\KVS@Global
496 }

```

\kv@set@family@handler

```

497 \long\def\kv@set@family@handler#1#2{%
498 \begingroup
499 \csname @safe@activetrue\endcsname
500 \let\ifincsname\iftrue
501 \expandafter\endgroup
502 \expandafter\def\csname KVS@#1@handler\endcsname##1##2{#2}%
503 }

```

\kv@unset@family@handler

```

504 \long\def\kv@unset@family@handler#1#2{%
505   \begingroup
506     \csname @safe@activetrue\endcsname
507     \let\ifincsname\iftrue
508   \expandafter\endgroup
509   \expandafter\let\csname KVS@#1@handler\endcsname\@UnDeFiNeD
510 }

```

3.9 Error handling

\kv@error@novalue

```

511 \def\kv@error@novalue{%
512   \kv@error@generic{No value specified for}%
513 }

```

\kv@error@unknownkey

```

514 \def\kv@error@unknownkey{%
515   \kv@error@generic{Undefined}%
516 }

```

\kv@error@generic

```

517 \def\kv@error@generic#1#2#3{%
518   \PackageError{kvsetkeys}{%
519     #1 key ‘#3’%
520   }{%
521     The keyval family of the key ‘#3’ is ‘#2’.\MessageBreak
522     The setting of the key is ignored because of the error.\MessageBreak
523     \MessageBreak
524     \@ehc
525   }%
526 }

```

3.10 Do it all

\kvsetkeys

```

527 \long\def\kvsetkeys#1#2{%
528   \kv@parse{#2}{\kv@processor@default{#1}}%
529 }

```

\kvsetkeys@expandafter

```

530 \def\kvsetkeys@expandafter#1#2{%
531   \expandafter\kv@parse\expandafter{#2}{%
532     \kv@processor@default{#1}%
533   }%
534 }

```

\KVS@cmd

```

535 \def\KVS@cmd{0}%

```

\KVS@cmd@inc

```

536 \def\KVS@cmd@inc{%
537   & \edef\KVS@cmd{\the\numexpr\KVS@cmd+1}%
538   $ \begingroup
539   $   \count255=\KVS@cmd\relax
540   $   \advance\count255 by 1\relax
541   $ \edef\x{\endgroup
542   $   \noexpand\def\noexpand\KVS@cmd{\number\count255}}%

```

```

543 $ }%
544 $ \x
545 }

\KVS@cmd@dec
546 \def\KVS@cmd@dec{%
547 & \edef\KVS@cmd{\the\numexpr\KVS@cmd-1}%
548 $ \begingroup
549 $ \count255=\KVS@cmd\relax
550 $ \advance\count255 by -1\relax
551 $ \edef\x{\endgroup
552 $ \noexpand\def\noexpand\KVS@cmd{\number\count255}%
553 $ }%
554 $ \x
555 }

```

\KVS@empty

```
556 \def\KVS@empty{}
```

\kvsetknownkeys

```

557 \def\kvsetknownkeys{%
558 \expandafter
559 \KVS@setknownkeys\csname KVS@cmd\KVS@cmd\endcsname{}%
560 }

```

\KVS@setknownkeys

```

561 \long\def\KVS@setknownkeys#1#2#3#4#5{%
562 \let#1\KVS@empty
563 \KVS@cmd@inc
564 #2\kv@parse#2{#5}{\kv@processor@known{#3}#1}%
565 \KVS@cmd@dec
566 \let#4=#1%
567 }

```

\kvsetknownkeys@expandafter

```

568 \def\kvsetknownkeys@expandafter{%
569 \expandafter
570 \KVS@setknownkeys
571 \csname KVS@cmd\KVS@cmd\endcsname\expandafter
572 }

573 \KVS@AtEnd%
574 \</package>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/kvsetkeys/kvsetkeys.dtx](https://ctan.org/ctan/packages/macros/latex/contrib/kvsetkeys/kvsetkeys.dtx) The source file.

[CTAN:macros/latex/contrib/kvsetkeys/kvsetkeys.pdf](https://ctan.org/ctan/packages/macros/latex/contrib/kvsetkeys/kvsetkeys.pdf) Documentation.

¹[CTAN:pkg/kvsetkeys](https://ctan.org/ctan/packages/pkg/kvsetkeys)

Bundle. All the packages of the bundle ‘kvsetkeys’ 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/kvsetkeys.tds.zip](#)

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

4.2 Bundle installation

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

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

4.3 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/kvsetkeys/kvsetkeys.sty
kvsetkeys.pdf      → doc/latex/kvsetkeys/kvsetkeys.pdf
kvsetkeys-example.tex → doc/latex/kvsetkeys/kvsetkeys-example.tex
kvsetkeys.dtx      → source/latex/kvsetkeys/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`.

4.4 Refresh file name databases

If your T_EX distribution (T_EX Live, MiK_TE_X, ...) relies on file name databases, you must refresh these. For example, T_EX Live users run `texhash` or `mktexlsr`.

4.5 Some details for the interested

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] A guide to key-value methods, Joseph Wright, second draft for TUGBoat, 2009-03-17. <https://www.texdev.net/uploads/2009/03/keyval.pdf>
- [2] David Carlisle: *The keyval package*; 1999/03/16 v1.13; CTAN:pkg/keyval.

6 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 -T_EX 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.

[2009/12/22 v1.7]

- Internal optimization (`\KVS@CommaSpace`, `\KVS@EqualsSpace`).

[2010/01/28 v1.8]

- Compatibility to `iniTeX` added.

[2010/03/01 v1.9]

- Support of `\par` inside values.

[2011/01/30 v1.10]

- Already loaded package files are not input in plain `TeX`.

[2011/03/03 v1.11]

- `\kv@break` and `\comma@break` added.

[2011/04/05 v1.12]

- Error message with recovery action in help message (request by GL).

[2011/04/07 v1.13]

- `\kv@processor@default` supports package `babel`’s shorthands.
- `\kv@set@family@handler` with shorthand support.

[2011/06/15 v1.14]

- Some optimizations in token register uses (GL, HO).

[2011/10/18 v1.15]

- `\kv@processor@known` and `\kvsetknownkeys` added.
- `\kvsetkeys@expandafter` and `\kvsetknownkeys@expandafter` added.
- Family handler can report success or failure by `\kv@handled@true` or `\kv@handled@false`.
- `\kv@unset@family@handler` added.

[2012/04/25 v1.16]

- `\kv@processor@default` and `\kv@processor@known` define macro `\kv@fam` for convenience.
- Catcode section: Catcode setting for + added for ε -TeX.

[2016/05/16 v1.17]

- Documentation updates.

[2019/12/15 v1.18]

- Documentation updates.
- Avoid `etexcmds` and `infwarwerr` in L^AT_EX.

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; plain numbers refer to the code lines where the entry is used.

Symbols	
<code>\\$</code>	178, 181
<code>\&</code>	179, 182
<code>\,</code>	225, 226
<code>\=</code>	36, 274, 275
<code>\></code>	38, 39, 40, 41, 42, 43
<code>\@PackageError</code>	171
<code>\@UnDeFiNeD</code>	509
<code>\@ehc</code>	524
<code>\@empty</code>	13
<code>\@endslash</code>	13, 16, 30
<code>\@undefined</code>	105, 177
<code>\@</code>	37, 38, 39, 40, 41, 42, 43, 44
<code>\~</code>	226, 275
A	
<code>\advance</code>	540, 550
<code>\aftergroup</code>	76
B	
<code>\begin</code>	34, 35
C	
<code>\catcode</code> .	49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 80, 81, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 116, 117, 119, 120, 121, 125, 126, 127, 128, 129, 130, 131, 134, 135, 137, 138, 139, 140, 144, 146, 178, 179, 181, 182
<code>\comma@break</code>	7, <u>385</u>
<code>\comma@entry</code>	376
<code>\comma@list</code>	222, 368
<code>\comma@normalize</code>	6, <u>213</u> , 367
<code>\comma@parse</code>	6, <u>366</u>
<code>\comma@parse@normalized</code> .	6, 368, <u>370</u>
<code>\count</code>	539, 540, 542, 549, 550, 552
<code>\csname</code>	61, 68, 97, 113, 123, 163, 166, 174, 397, 408, 411, 415, 426, 429, 433, 439, 450, 453, 457, 468, 471, 475, 499, 502, 506, 509, 559, 571
D	
<code>\define@key</code>	29
<code>\documentclass</code>	2
<code>\dots</code>	38, 41, 43
E	
<code>\empty</code>	64, 65
<code>\end</code>	45, 46
<code>\endcsname</code>	61, 68, 97, 113, 123, 163, 166, 174, 397, 406, 408, 409, 411, 415, 424, 426, 429, 433, 439, 448, 450, 451, 453, 457, 466, 468, 471, 475, 499, 502, 506, 509, 559, 571
<code>\endinput</code>	76, 161
<code>\endlinechar</code>	51, 82, 118, 124, 136
I	
<code>\ifcsname</code> .	406, 409, 424, 448, 451, 466
<code>\iffalse</code>	389
<code>\ifincsname</code>	398, 440, 500, 507
<code>\ifkv@handled@</code>	389, 392, <u>394</u> , 417, 459
<code>\iftrue</code>	392, 398, 440, 500, 507
<code>\ifx</code>	62, 65, 68, 97, 105, 108, 163, 166, 174,

	279, 282, 289, 291, 301, 307, 313, 317, 353, 354, 489, 491, 493		
<code>\toksdef</code>	175	
			W
			<code>\write</code>
			70, 99
			U
<code>\unexpanded</code>	188, 351, 483, 485	
<code>\unless</code>	...	406, 409, 424, 448, 451, 466	
<code>\usepackage</code>	3, 4, 5	
			X
<code>\x</code>	...	61, 62, 65, 69, 73, 75, 98, 103, 113, 122, 134, 541, 544, 551, 554	