

# The kvsetkeys package

Heiko Oberdiek  
<heiko.oberdiek at googlemail.com>

2012/04/25 v1.16

## 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 . . . . .	3
1.3	Parsing key value lists . . . . .	3
1.4	Processing key value pairs . . . . .	4
1.4.1	Processing similar to <code>keyval</code> . . . . .	4
1.4.2	Processing similar to <code>\setkeys*</code> of package <code>xkeyval</code> . . . . .	4
1.5	Default family handler . . . . .	5
1.6	Put it all together . . . . .	5
1.7	Comma separated lists . . . . .	6
<b>2</b>	<b>Example</b>	<b>6</b>
<b>3</b>	<b>Implementation</b>	<b>7</b>
3.1	Identification . . . . .	7
3.2	Package loading . . . . .	9
3.3	Check for $\epsilon$ -TeX . . . . .	10
3.4	Generic help macros . . . . .	10
3.5	Normalizing . . . . .	10
3.6	Parsing key value lists . . . . .	13
3.7	Parsing comma lists . . . . .	14
3.8	Processing key value pairs . . . . .	14
3.9	Error handling . . . . .	17
3.10	Do it all . . . . .	17
<b>4</b>	<b>Test</b>	<b>18</b>
4.1	Catcode checks for loading . . . . .	18
4.2	Macro tests . . . . .	20
4.2.1	Preamble . . . . .	20
4.2.2	Time . . . . .	20
4.2.3	Test sets . . . . .	21
4.3	Tests for key value processing handler . . . . .	24

<b>5</b>	<b>Installation</b>	<b>25</b>
5.1	Download	25
5.2	Bundle installation	25
5.3	Package installation	25
5.4	Refresh file name databases	26
5.5	Some details for the interested	26
<b>6</b>	<b>Catalogue</b>	<b>27</b>
<b>7</b>	<b>References</b>	<b>27</b>
<b>8</b>	<b>History</b>	<b>27</b>
	[2006/03/06 v1.0]	27
	[2006/10/19 v1.1]	27
	[2007/09/09 v1.2]	27
	[2007/09/29 v1.3]	28
	[2009/07/19 v1.4]	28
	[2009/07/30 v1.5]	28
	[2009/12/12 v1.6]	28
	[2009/12/22 v1.7]	28
	[2010/01/28 v1.8]	28
	[2010/03/01 v1.9]	28
	[2011/01/30 v1.10]	28
	[2011/03/03 v1.11]	28
	[2011/04/05 v1.12]	28
	[2011/04/07 v1.13]	28
	[2011/06/15 v1.14]	28
	[2011/10/18 v1.15]	29
	[2012/04/25 v1.16]	29
<b>9</b>	<b>Index</b>	<b>29</b>

# 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}}}}

```

→ value  
→ {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 2012/04/25 v1.16 `\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{\langle family \rangle}{\langle cmd \rangle}{\langle key \rangle}{\langle value \rangle}`

The key value processor `\kv@processor@known` behaves similar to `\kv@processor@default`. If the `\langle key \rangle` exists in the `\langle family \rangle` 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 `\langle cmd \rangle`. Since 2011/10/18 v1.15.

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
    handled = true
     $\langle handler \rangle \{ \langle key \rangle \} \{ \langle value \rangle \}$ 
    if handled
    else
      add " $\{ \langle key \rangle \} = \{ \langle value \rangle \}$ " to  $\{ \langle cmd \rangle \}$ 
    fi
  else
    add " $\{ \langle key \rangle \} = \{ \langle value \rangle \}$ " to  $\{ \langle cmd \rangle \}$ 
    raise unknown key error
  fi
fi

```

Since 2012/04/25 v1.16 `\kv@processor@known` also defines macro `\kv@fam` with meaning  $\langle family \rangle$  for convenience.

## 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`.

```
\kv@unset@family@handler  $\{ \langle family \rangle \}$ 
```

It removes the family handler for  $\langle family \rangle$ . Since 2011/10/18 v1.15.

## 1.6 Put it all together

```
\kvsetkeys  $\{ \langle family \rangle \} \{ \langle key\ value\ list \rangle \}$ 
```

Macro `\kvsetkeys` processes the  $\langle key\ value\ list \rangle$  with the standard processor `\kv@processor@default`:

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

```
\kvsetknownkeys  $\{ \langle family \rangle \} \{ \langle cmd \rangle \} \{ \langle key\ value\ list \rangle \}$ 
```

Macro `\kvsetknownkeys` processes the  $\langle key\ value\ list \rangle$  with processor `\kv@processor@known`. All key value pairs with keys that are not known in  $\langle family \rangle$  are stored in macro  $\langle cmd \rangle$ . A previous contents of macro  $\langle cmd \rangle$  will be overwritten. If all keys can be handled,  $\langle cmd \rangle$  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  $\langle cmdaux \rangle$  with unique name (inside the current group)
\def $\langle cmdaux \rangle \{ \}$ 

```

```
\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`.

```
\comma@break
```

Since version 2011/03/03 v1.11 `\comma@break` can be called inside the `\processor` 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 \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\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 [2012/04/25 v1.16 Key value parser (H0)]%
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   \TMP@RequirePackage{etexcmds}[2010/01/28]%
172 \else
173   \RequirePackage{infwarerr}[2007/09/09]%
174   \RequirePackage{etexcmds}[2010/01/28]%
175 \fi

176 \expandafter\ifx\csname toks@\endcsname\relax
177   \toksdef\toks@=0 %
178 \fi

```

### 3.3 Check for -TeX

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

```

179 \begingroup\expandafter\endgroup
180 \ifcase0\ifetex@unexpanded
181     \expandafter\ifx\csname ifcsname\endcsname\relax
182     \else
183     \expandafter\ifx\csname unless\endcsname\relax
184     \else
185     1%
186     \fi
187     \fi
188     \fi
189 \catcode`\$=9 % ignore
190 \catcode`\&=14 % comment
191 \else % e-TeX
192 \catcode`\$=14 % comment
193 \catcode`\&=9 % ignore
194 \fi

```

### 3.4 Generic help macros

```

\KVS@Empty
195 \def\KVS@Empty{}

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

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

\KVS@IfEmpty
198 \long\def\KVS@IfEmpty#1{%
199 & \edef\KVS@Temp{\etex@unexpanded{#1}}%
200 $ \begingroup
201 $ \toks@{#1}%
202 $ \edef\KVS@Temp{\the\toks@}%
203 $ \expandafter\endgroup
204 \ifx\KVS@Temp\KVS@Empty
205 \expandafter\KVS@FirstOfTwo
206 \else
207 \expandafter\KVS@SecondOfTwo
208 \fi
209 }

```

### 3.5 Normalizing

```

\kv@normalize
210 \long\def\kv@normalize#1{%
211 \begingroup
212 \toks@{, #1,}%
213 \KVS@Comma
214 \KVS@SpaceComma
215 \KVS@CommaSpace
216 \KVS@CommaComma
217 \KVS@Equals
218 \KVS@SpaceEquals
219 \KVS@EqualsSpace
220 \xdef\KVS@Global{\the\toks@}%
221 \endgroup
222 \let\kv@list\KVS@Global
223 }

```

\comma@normalize

```

224 \def\comma@normalize#1{%
225   \begingroup
226     \toks@{,#1,}%
227     \KVS@Comma
228     \KVS@SpaceComma
229     \KVS@CommaSpace
230     \KVS@CommaComma
231     \xdef\KVS@Global{\the\toks@}%
232   \endgroup
233   \let\comma@list\KVS@Global
234 }

```

\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.

```

235 \begingroup
236   \lccode`\,=\,%
237   \lccode`\~=\,%
238 \lowercase{\endgroup
239   \def\KVS@Comma{%
240     \toks@\expandafter{\expandafter}\expandafter
241     \KVS@@Comma\the\toks@~\KVS@Nil
242   }%
243   \long\def\KVS@@Comma#1~#2\KVS@Nil{%
244     \toks@\expandafter{\the\toks@#1}%
245     \KVS@IfEmpty{#2}{%
246       }{%
247         \KVS@@Comma,#2\KVS@Nil
248       }%
249     }%
250 }

```

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

```

251 \def\KVS@SpaceComma#1{%
252   \def\KVS@SpaceComma{%
253     \expandafter\KVS@@SpaceComma\the\toks@#1,\KVS@Nil
254   }%
255 }
256 \KVS@SpaceComma{ }

```

\KVS@@SpaceComma

```

257 \long\def\KVS@@SpaceComma#1 ,#2\KVS@Nil{%
258   \KVS@IfEmpty{#2}{%
259     \toks@{#1}%
260   }{%
261     \KVS@@SpaceComma#1,#2\KVS@Nil
262   }%
263 }

```

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

```

264 \def\KVS@CommaSpace{%
265   \expandafter\KVS@@CommaSpace\the\toks@, \KVS@Nil
266 }

```

\KVS@@CommaSpace

```

267 \long\def\KVS@@CommaSpace#1 ,#2\KVS@Nil{%
268   \KVS@IfEmpty{#2}{%
269     \toks@{#1}%
270   }{%
271     \KVS@@CommaSpace#1,#2\KVS@Nil
272   }%
273 }

```

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

```

274 \def\KVS@CommaComma{%
275   \expandafter\KVS@@CommaComma\the\toks@,\KVS@Nil
276 }

```

`\KVS@@CommaComma`

```

277 \long\def\KVS@@CommaComma#1,,#2\KVS@Nil{%
278   \KVS@IfEmpty{#2}{%
279     \toks@{#1},}% (!)
280   }{%
281     \KVS@@CommaComma#1,#2\KVS@Nil
282   }%
283 }

```

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

```

284 \begingroup
285   \lccode`\==`\=%
286   \lccode`\~=\`=%
287 \lowercase{\endgroup
288   \def\KVS@Equals{%
289     \toks@\expandafter{\expandafter}\expandafter
290     \KVS@@Equals\the\toks@~\KVS@Nil
291   }%
292   \long\def\KVS@@Equals#1~#2\KVS@Nil{%
293     \edef\KVS@Temp{\the\toks@}%
294     \ifx\KVS@Temp\KVS@Empty
295       \expandafter\KVS@FirstOfTwo
296     \else
297       \expandafter\KVS@SecondOfTwo
298     \fi
299     {%
300       \toks@{#1}%
301     }{%
302       \toks@\expandafter{\the\toks@=#1}%
303     }%
304     \KVS@IfEmpty{#2}{%
305     }{%
306       \KVS@@Equals#2\KVS@Nil
307     }%
308   }%
309 }

```

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

```

310 \def\KVS@SpaceEquals#1{%
311   \def\KVS@SpaceEquals{%
312     \expandafter\KVS@@SpaceEquals\the\toks@#1=\KVS@Nil
313   }%
314 }
315 \KVS@SpaceEquals{ }

```

`\KVS@@SpaceEquals`

```

316 \long\def\KVS@@SpaceEquals#1=#2\KVS@Nil{%
317   \KVS@IfEmpty{#2}{%
318     \toks@{#1}%
319   }{%
320     \KVS@@SpaceEquals#1=#2\KVS@Nil
321   }%
322 }

```

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

```

323 \def\KVS@EqualsSpace{%
324   \expandafter\KVS@@EqualsSpace\the\toks@= \KVS@Nil
325 }

```

\KVS@@EqualsSpace

```

326 \long\def\KVS@@EqualsSpace#1= #2\KVS@Nil{%
327   \KVS@IfEmpty{#2}{%
328     \toks@{#1}%
329   }{%
330     \KVS@@EqualsSpace#1=#2\KVS@Nil
331   }%
332 }
```

### 3.6 Parsing key value lists

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

```

333 \long\def\kv@parse#1{%
334   \kv@normalize{#1}%
335   \expandafter\kv@parse@normalized\expandafter{\kv@list}%
336 }
```

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

#2: processor

```

337 \long\def\kv@parse@normalized#1#2{%
338   \KVS@Parse#1,\KVS@Nil{#2}%
339 }
```

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

#3: processor

```

340 \long\def\KVS@Parse#1,#2\KVS@Nil#3{%
341   \KVS@IfEmpty{#1}{%
342   }{%
343     \KVS@Process#1=\KVS@Nil{#3}%
344   }%
345   \KVS@MaybeBreak
346   \KVS@IfEmpty{#2}{%
347   }{%
348     \KVS@Parse#2\KVS@Nil{#3}%
349   }%
350 }
```

\KVS@Process #1: key

#2: value, =

#3: processor

```

351 \long\def\KVS@Process#1=#2\KVS@Nil#3{%
352   \let\KVS@MaybeBreak\relax
353   \def\kv@key{#1}%
354   \KVS@IfEmpty{#2}{%
355     \let\kv@value\relax
356     #3{#1}{}%
357   }{%
358     \KVS@@Process{#1}#2\KVS@Nil{#3}%
359   }%
360 }
```

\KVS@@Process #1: key

#2: value

#3: processor

```

361 \long\def\KVS@@Process#1#2=\KVS@Nil#3{%
362   & \edef\kv@value{\etex@unexpanded{#2}}%
363   $ \begingroup
364   $   \toks@{#2}%
365   $   \xdef\KVS@Global{\the\toks@}%
366   $ \endgroup
367   $ \let\kv@value\KVS@Global
```

```

368   #3{#1}{#2}%
369 }

\KVS@MaybeBreak

370 \let\KVS@MaybeBreak\relax

\KVS@break

371 \def\KVS@break#1#2#3#4{%
372   \let\KVS@MaybeBreak\relax
373 }

\kv@break

374 \def\kv@break{%
375   \let\KVS@MaybeBreak\KVS@break
376 }

```

### 3.7 Parsing comma lists

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

```

377 \def\comma@parse#1{%
378   \comma@normalize{#1}%
379   \expandafter\comma@parse@normalized\expandafter{\comma@list}%
380 }

```

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

```

381 \def\comma@parse@normalized#1#2{%
382   \KVS@CommaParse#1,\KVS@Nil{#2}%
383 }

```

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

```

384 \def\KVS@CommaParse#1,#2\KVS@Nil#3{%
385   \KVS@IfEmpty{#1}{%
386   }{%
387     \def\comma@entry{#1}%
388     #3{#1}%
389   }%
390   \KVS@MaybeBreak
391   \KVS@IfEmpty{#2}{%
392   }{%
393     \KVS@CommaParse#2\KVS@Nil{#3}%
394   }%
395 }

```

`\comma@break`

```

396 \def\comma@break{%
397   \let\KVS@MaybeBreak\KVS@break
398 }

```

### 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).

```

399 \def\kv@handled@false{%
400   \let\ifkv@handled@iffalse
401 }

```

```

\kv@handled@true
402 \def\kv@handled@true{%
403   \let\ifkv@handled@\iftrue
404 }

\ifkv@handled@
405 \kv@handled@true

\kv@processor@default
406 \def\kv@processor@default#1#2{%
407   \begingroup
408     \csname @safe@activetrue\endcsname
409     \let\ifincsname\iftrue
410     \edef\KVS@temp{\endgroup
411       \noexpand\KVS@ProcessorDefault{#1}{#2}%
412     }%
413   \KVS@temp
414 }

\KVS@ProcessorDefault
415 \long\def\KVS@ProcessorDefault#1#2#3{%
416   \def\kv@fam{#1}%
417   & \unless\ifcsname KV@#1@#2\endcsname
418   $ \begingroup\expandafter\expandafter\expandafter\endgroup
419   $ \expandafter\ifx\csname KV@#1@#2\endcsname\relax
420   & \unless\ifcsname KVS@#1@handler\endcsname
421   $ \begingroup\expandafter\expandafter\expandafter\endgroup
422   $ \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
423     \kv@error@unknownkey{#1}{#2}%
424   \else
425     \kv@handled@true
426     \csname KVS@#1@handler\endcsname{#2}{#3}%
427     \relax
428     \ifkv@handled@
429     \else
430       \kv@error@unknownkey{#1}{#2}%
431     \fi
432   \fi
433   \else
434     \ifx\kv@value\relax
435     & \unless\ifcsname KV@#1@#2@default\endcsname
436     $ \begingroup\expandafter\expandafter\expandafter\endgroup
437     $ \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
438       \kv@error@novalue{#1}{#2}%
439     \else
440       \csname KV@#1@#2@default\endcsname
441       \relax
442     \fi
443   \else
444     \csname KV@#1@#2\endcsname{#3}%
445   \fi
446 \fi
447 }

\kv@processor@known
448 \def\kv@processor@known#1#2#3{%
449   \begingroup
450     \csname @safe@activetrue\endcsname
451     \let\ifincsname\iftrue
452     \edef\KVS@temp{\endgroup
453       \noexpand\KVS@ProcessorKnown{#1}\noexpand#2{#3}%
454     }%

```

```

455 \KVS@temp
456 }

```

\KVS@ProcessorKnown

```

457 \long\def\KVS@ProcessorKnown#1#2#3#4{%
458 \def\kv@fam{#1}%
459 & \unless\ifcsname KV@#1@#3\endcsname
460 $ \begingroup\expandafter\expandafter\expandafter\endgroup
461 $ \expandafter\ifx\csname KV@#1@#3\endcsname\relax
462 & \unless\ifcsname KVS@#1@handler\endcsname
463 $ \begingroup\expandafter\expandafter\expandafter\endgroup
464 $ \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
465 \KVS@AddUnhandled#2{#3}{#4}%
466 \else
467 \kv@handled@true
468 \csname KVS@#1@handler\endcsname{#3}{#4}%
469 \relax
470 \ifkv@handled@
471 \else
472 \KVS@AddUnhandled#2{#3}{#4}%
473 \fi
474 \fi
475 \else
476 \ifx\kv@value\relax
477 & \unless\ifcsname KV@#1@#2@default\endcsname
478 $ \begingroup\expandafter\expandafter\expandafter\endgroup
479 $ \expandafter\ifx\csname KV@#1@#3@default\endcsname\relax
480 \kv@error@novalue{#1}{#3}%
481 \else
482 \csname KV@#1@#3@default\endcsname
483 \relax
484 \fi
485 \else
486 \csname KV@#1@#3\endcsname{#4}%
487 \fi
488 \fi
489 }

```

\KVS@AddUnhandled

```

490 \long\def\KVS@AddUnhandled#1#2#3{%
491 & \edef#1{%
492 & \ifx#1\KVS@empty
493 & \else
494 & \etex@unexpanded{#1},%
495 & \fi
496 & \etex@unexpanded{{#2}={#3}}}%
497 & }%
498 $ \begingroup
499 $ \ifx#1\KVS@empty
500 $ \toks@{{#2}={#3}}}%
501 $ \else
502 $ \toks@\expandafter{#1,{#2}={#3}}}%
503 $ \fi
504 $ \xdef\KVS@Global{\the\toks@}%
505 $ \endgroup
506 $ \let#1\KVS@Global
507 }

```

\kv@set@family@handler

```

508 \long\def\kv@set@family@handler#1#2{%
509 \begingroup
510 \csname @safe@activetrue\endcsname

```

```

511 \let\ifincsname\iftrue
512 \expandafter\endgroup
513 \expandafter\def\csname KVS@#1@handler\endcsname##1##2{#2}%
514 }

```

\kv@unset@family@handler

```

515 \long\def\kv@unset@family@handler#1#2{%
516 \begingroup
517 \csname @safe@activetrue\endcsname
518 \let\ifincsname\iftrue
519 \expandafter\endgroup
520 \expandafter\let\csname KVS@#1@handler\endcsname\@UnDeFiNeD
521 }

```

### 3.9 Error handling

\kv@error@novalue

```

522 \def\kv@error@novalue{%
523 \kv@error@generic{No value specified for}%
524 }

```

\kv@error@unknownkey

```

525 \def\kv@error@unknownkey{%
526 \kv@error@generic{Undefined}%
527 }

```

\kv@error@generic

```

528 \def\kv@error@generic#1#2#3{%
529 \@PackageError{kvsetkeys}{%
530 #1 key `#3'%
531 }{%
532 The keyval family of the key `#3' is `#2'.\MessageBreak
533 The setting of the key is ignored because of the error.\MessageBreak
534 \MessageBreak
535 \@ehc
536 }%
537 }

```

### 3.10 Do it all

\kvsetkeys

```

538 \long\def\kvsetkeys#1#2{%
539 \kv@parse{#2}{\kv@processor@default{#1}}%
540 }

```

\kvsetkeys@expandafter

```

541 \def\kvsetkeys@expandafter#1#2{%
542 \expandafter\kv@parse\expandafter{#2}{%
543 \kv@processor@default{#1}%
544 }%
545 }

```

\KVS@cmd

```

546 \def\KVS@cmd{0}%

```

\KVS@cmd@inc

```

547 \def\KVS@cmd@inc{%
548 & \edef\KVS@cmd{\the\numexpr\KVS@cmd+1}%
549 $ \begingroup
550 $ \count255=\KVS@cmd\relax
551 $ \advance\count255 by 1\relax

```

```

552 $ \edef\x{\endgroup
553 $ \noexpand\def\noexpand\KVS@cmd{\number\count255}%
554 $ }%
555 $ \x
556 }

```

\KVS@cmd@dec

```

557 \def\KVS@cmd@dec{%
558 & \edef\KVS@cmd{\the\numexpr\KVS@cmd-1}%
559 $ \begingroup
560 $ \count255=\KVS@cmd\relax
561 $ \advance\count255 by -1\relax
562 $ \edef\x{\endgroup
563 $ \noexpand\def\noexpand\KVS@cmd{\number\count255}%
564 $ }%
565 $ \x
566 }

```

\KVS@empty

```

567 \def\KVS@empty{}

```

\kvsetknownkeys

```

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

```

\KVS@setknownkeys

```

572 \long\def\KVS@setknownkeys#1#2#3#4#5{%
573 \let#1\KVS@empty
574 \KVS@cmd@inc
575 #2\kv@parse#2{#5}{\kv@processor@known{#3}#1}%
576 \KVS@cmd@dec
577 \let#4=#1%
578 }

```

\kvsetknownkeys@expandafter

```

579 \def\kvsetknownkeys@expandafter{%
580 \expandafter
581 \KVS@setknownkeys
582 \csname KVS@cmd\KVS@cmd\endcsname\expandafter
583 }

584 \KVS@AtEnd%
585 </package>

```

## 4 Test

### 4.1 Catcode checks for loading

```

586 <*test1>

587 \catcode`\{=1 %
588 \catcode`\}=2 %
589 \catcode`\#=6 %
590 \catcode`\@=11 %
591 \expandafter\ifx\csname count@\endcsname\relax
592 \countdef\count@=255 %
593 \fi
594 \expandafter\ifx\csname @gobble\endcsname\relax
595 \long\def@gobble#1{}%
596 \fi

```

```

597 \expandafter\ifx\csname @firstofone\endcsname\relax
598 \long\def\@firstofone#1{#1}%
599 \fi
600 \expandafter\ifx\csname loop\endcsname\relax
601 \expandafter\@firstofone
602 \else
603 \expandafter\@gobble
604 \fi
605 {%
606 \def\loop#1\repeat{%
607 \def\body{#1}%
608 \iterate
609 }%
610 \def\iterate{%
611 \body
612 \let\next\iterate
613 \else
614 \let\next\relax
615 \fi
616 \next
617 }%
618 \let\repeat=\fi
619 }%
620 \def\RestoreCatcodes{}
621 \count@=0 %
622 \loop
623 \edef\RestoreCatcodes{%
624 \RestoreCatcodes
625 \catcode\the\count@=\the\catcode\count@\relax
626 }%
627 \ifnum\count@<255 %
628 \advance\count@ 1 %
629 \repeat
630
631 \def\RangeCatcodeInvalid#1#2{%
632 \count@=#1\relax
633 \loop
634 \catcode\count@=15 %
635 \ifnum\count@<#2\relax
636 \advance\count@ 1 %
637 \repeat
638 }
639 \def\RangeCatcodeCheck#1#2#3{%
640 \count@=#1\relax
641 \loop
642 \ifnum#3=\catcode\count@
643 \else
644 \errmessage{%
645 Character \the\count@\space
646 with wrong catcode \the\catcode\count@\space
647 instead of \number#3%
648 }%
649 \fi
650 \ifnum\count@<#2\relax
651 \advance\count@ 1 %
652 \repeat
653 }
654 \def\space{ }
655 \expandafter\ifx\csname LoadCommand\endcsname\relax
656 \def\LoadCommand{\input kvsetkeys.sty\relax}%
657 \fi
658 \def\Test{%

```

```

659 \RangeCatcodeInvalid{0}{47}%
660 \RangeCatcodeInvalid{58}{64}%
661 \RangeCatcodeInvalid{91}{96}%
662 \RangeCatcodeInvalid{123}{255}%
663 \catcode`\@=12 %
664 \catcode`\=0 %
665 \catcode`\%=14 %
666 \LoadCommand
667 \RangeCatcodeCheck{0}{36}{15}%
668 \RangeCatcodeCheck{37}{37}{14}%
669 \RangeCatcodeCheck{38}{47}{15}%
670 \RangeCatcodeCheck{48}{57}{12}%
671 \RangeCatcodeCheck{58}{63}{15}%
672 \RangeCatcodeCheck{64}{64}{12}%
673 \RangeCatcodeCheck{65}{90}{11}%
674 \RangeCatcodeCheck{91}{91}{15}%
675 \RangeCatcodeCheck{92}{92}{0}%
676 \RangeCatcodeCheck{93}{96}{15}%
677 \RangeCatcodeCheck{97}{122}{11}%
678 \RangeCatcodeCheck{123}{255}{15}%
679 \RestoreCatcodes
680 }
681 \Test
682 \csname @@end\endcsname
683 \end
684 </test1>

```

## 4.2 Macro tests

### 4.2.1 Preamble

```

685 <*test2>
686 \NeedsTeXFormat{LaTeX2e}
687 \nofiles
688 \documentclass{article}
689 <noetex> \let\SavedUnexpanded\unexpanded
690 <noetex> \let\unexpanded\UNDEFINED
691 \makeatletter
692 \chardef\KVS@TestMode=1 %
693 \makeatother
694 \usepackage{kvsetkeys}[2012/04/25]
695 <noetex> \let\unexpanded\SavedUnexpanded
696 \usepackage{qstest}
697 \IncludeTests{*}
698 \LogTests{log}{*}{*}

```

### 4.2.2 Time

```

699 \begingroup\expandafter\expandafter\expandafter\endgroup
700 \expandafter\ifx\csname pdfresettimer\endcsname\relax
701 \else
702 \makeatletter
703 \newcount\SummaryTime
704 \newcount\TestTime
705 \SummaryTime=\z@
706 \newcommand*\PrintTime[2]{%
707 \typeout{%
708 [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
709 }%
710 }%
711 \newcommand*\StartTime[1]{%
712 \renewcommand*\TimeDescription{#1}%
713 \pdfresettimer
714 }%

```

```

715 \newcommand*{\TimeDescription}{}%
716 \newcommand*{\StopTime}{}%
717 \TestTime=\pdfelapsedtime
718 \global\advance\SummaryTime\TestTime
719 \PrintTime\TimeDescription\TestTime
720 }%
721 \let\saved@qstest\qstest
722 \let\saved@endqstest\endqstest
723 \def\qstest#1#2{%
724 \saved@qstest{#1}{#2}%
725 \StartTime{#1}%
726 }%
727 \def\endqstest{%
728 \StopTime
729 \saved@endqstest
730 }%
731 \AtEndDocument{%
732 \PrintTime{summary}\SummaryTime
733 }%
734 \makeatother
735 \fi

```

#### 4.2.3 Test sets

```

736 \makeatletter
737 \def\@makeactive#1{%
738 \catcode`#1=13\relax
739 }
740 \@makeactive\,
741 \def,{\errmessage{COMMA}}
742 \@makeother\,
743 \@makeactive\=
744 \def={\errmessage{EQUALS}}
745 \@makeother\=
746
747 \begin{qstest}{normalize}{normalize,active-chars,space-removal}%
748 \long\def\Test#1#2{%
749 \@makeother\,%
750 \@makeother\=%
751 \scantokens{\toks@={#2}}%
752 \edef\Result{\the\toks@}%
753 \@makeother\,%
754 \@makeother\=%
755 \@Test{#1}%
756 \@makeactive\,%
757 \@Test{#1}%
758 \@makeactive\=%
759 \@Test{#1}%
760 \@makeother\,%
761 \@Test{#1}%
762 \@makeother\=%
763 }%
764 \long\def\@Test#1{%
765 \scantokens{\kv@normalize{#1}}%
766 \expandafter\expandafter\expandafter\Expect
767 \expandafter\expandafter\expandafter
768 {\expandafter\kv@list\expandafter}\expandafter{\Result}%
769 \Expect*{\ifx\kv@list\Result true\else false\fi}{true}%
770 }%
771 \Test{}{,%
772 \Test{,}{,%
773 \Test{,,}{,%
774 \Test{,,,}{,%
775 \Test{ , }{,%

```

```

776 \Test{{a}}{,{a},}%
777 \Test{,{a}}{,{a},}%
778 \Test{{a},}{,{a},}%
779 \Test{{a},{b}}{,{a},{b},}%
780 \Test{{b}={c},{},{}={},{d}={},{b}={c},{},{}={},{d}={},}%
781 \Test{{}}{,{},}%
782 \Test{{},{},{}}{,{},{},{},}%
783 \Test{=}{,{},}%
784 \Test{=,=}{,{},}%
785 \Test{a=\par}{,a=\par,}%
786 \Test{\par}{,\par,}%
787 \def\TestSet#1{%
788   \Test{#1#1}{,%
789     \Test{#1#1,#1#1}{,%
790       \Test{#1#1,#1#1,#1#1}{,%
791         \Test{#1#1#1#1#1}{,%
792           \Test{{a}#1#1=#1#1{b}}{,{a}={b},}%
793         }%
794       \TestSet{ }%
795     \begingroup
796       \let\saved@normalize\kv@normalize
797       \def\kv@normalize#1{%
798         \saved@normalize{#1}%
799         \@onelevel@sanitize\kv@list
800         \@onelevel@sanitize\Result
801       }%
802       \Test{#,#=#,{#}={#},{#}={#},{#}={#},{#}={#},{#}={#},{#}={#},}%
803     \endgroup
804   \begingroup
805     \def\Test#1#2{%
806       \edef\Result{#2}%
807       \@Test{#1}%
808     }%
809     \Test{{ a = b }}{,{ a = b },}%
810     \@makeactive\,%
811     \Test{{,}}{\string,{\noexpand,}\string,}%
812     \@makeother\,%
813     \@makeactive\=%
814     \Test{a={}}{,a\string={\noexpand=},}%
815   \endgroup
816   \Test{a=b}{,a=b,}%
817   \Test{a={b}}{,a={b},}%
818   \Test{a = {b}}{,a={b},}%
819   \Test{a= {b}}{,a={b},}%
820   \Test{a = {b}}{,a={b},}%
821   \Test{a = {b} ,}{,a={b},}%
822   \Test{a}{,a,}%
823   \Test{ a}{,a,}%
824   \Test{a }{,a,}%
825   \Test{ a }{,a,}%
826   \Test{, a ,}{,a,}%
827   \Test{, a b ,}{,a b ,}%
828   \Test{,a ,}{,a,}%
829   \Test{ a =}{,a=,}%
830   \Test{ a = }{,a=,}%
831   \Test{a =}{,a=,}%
832   \Test{{a} =}{,{a}=,}%
833   \Test{{a}= {} }{,{a}={},}%
834   \Test{, a = {} }{,a={},}%
835   \Test{a, b}{,a, b,}%
836   \Test{a=\fi}{,a=\fi,}%
837   \Test{a=\iffalse}{,a=\iffalse,}%

```

```

838 \Test{a=\iffalse,b=\fi}{,a=\iffalse,b=\fi,}%
839 \end{qstest}
840
841 \begin{qstest}{parse}{parse,brace-removal}
842 \def\Processor#1#2{%
843 \expandafter\Expect\expandafter{\kv@key}{#1}%
844 \toks@{#2}%
845 \edef\x{\the\toks@}%
846 \ifx\kv@value\relax
847 \Expect*{\the\toks@}{}%
848 \def\Value{<>}%
849 \else
850 \edef\Value{[\the\toks@]}%
851 \@onelevel@sanitize\Value
852 \fi
853 \toks@{#1}%
854 \ifx\Result\@empty
855 \edef\Result{[\the\toks@]=\Value}%
856 \else
857 \edef\Result{\Result,[\the\toks@]=\Value}%
858 \fi
859 \@onelevel@sanitize\Result
860 }%
861 \def\Test#1#2{%
862 \sbox0{%
863 \let\Result\@empty
864 \kv@parse{#1}\Processor
865 \Expect*{\Result}{#2}%
866 }%
867 \Expect*{\the\wd0}{0.0pt}%
868 }%
869 \Test{}{}%
870 \Test{{}}{}%
871 \Test{{{}}}{}=<>%
872 \Test{{{}}}{}[{}]=<>%
873 \Test{a}{[a]=<>}%
874 \Test{{a}}{[a]=<>}%
875 \Test{{a}}{[a]=<>}%
876 \Test{{{a}}}{}[a]=<>%
877 \Test{{{a}}}{}[a]=<>%
878 \Test{a=}{[a]=[]}%
879 \Test{{a}=}{[a]=[]}%
880 \Test{{{a}}}={}[a]=[]%
881 \Test{a={}}{[a]=[]}%
882 \Test{{a}={}}{[a]=[{}]}%
883 \Test{a=b}{[a]=[b]}%
884 \Test{a=\fi}{[a]=[\fi]}%
885 \Test{a=\iffalse}{[a]=[\iffalse]}%
886 \Test{a=\iffalse,b=\fi}{[a]=[\iffalse],[b]=[\fi]}%
887 \Test{{ a = b }}{[ a ]=[ b ]}%
888 \Test{{{ a = b }}}{[ a = b ]=<>}%
889 \end{qstest}
890
891 \begin{qstest}{comma}{comma,parse}
892 \def\Processor#1{%
893 \expandafter\Expect\expandafter{\comma@entry}{#1}%
894 \toks@{#1}%
895 \ifx\Result\@empty
896 \edef\Result{[\the\toks@]}%
897 \else
898 \edef\Result{\Result,[\the\toks@]}%
899 \fi

```

```

900 \@onelevel@sanitizet\Result
901 }%
902 \def\Test#1#2{%
903 \sbox0{%
904 \let\Result\@empty
905 \comma@parse{#1}\Processor
906 \Expect*{\Result}{#2}%
907 }%
908 \Expect*{\the\wd0}{0.0pt}%
909 }%
910 \Test{}{}%
911 \Test{{}}{}%
912 \Test{{{}}}{{{}}}%
913 \Test{a}{[a]}%
914 \Test{{a}}{{[a]}%
915 \Test{{{a}}}{{{[a]}}}%
916 \Test{a=}{[a=]}%
917 \Test{a\fi}{[a\fi]}%
918 \Test{a\iffalse}{[a\iffalse]}%
919 \Test{\iffalse,\fi}{[\iffalse],[\fi]}%
920 \Test{ a , b , c }{[a],[b],[c]}%
921 \Test{ { } , { } , { } , { } }{[ ],[ ],[ ],[ ]}%
922 \Test{ {{}} , {{}} , {{}} , {{}} }{[{}],[{}],[{}],[{}]}%
923 \end{qstest}
924
925 \begin{document}
926 \end{document}
927 </test2>

```

### 4.3 Tests for key value processing handler

```

928 <*test4>
929 \catcode`\{=1
930 \catcode`\}=2
931 \catcode`\#=6
932 \catcode`\@=11
933 \input kvdefinekeys.sty\relax
934 \input kvsetkeys.sty\relax
935 \input infwarerr.sty\relax
936 \def\Error#1{%
937 \PackageError{test}{#1}\@ehc
938 }

939 \def\temp#1#2{%
940 \kv@define@key{#1}{#2}{%
941 \edef\result{%
942 \result
943 [#1:#2=##1]} hash-ok
944 }%
945 }%
946 }

947 \temp{FA}{key1}
948 \temp{FA}{key2}
949 \temp{FB}{key3}
950 \temp{FB}{key3}

951 \setbox0=\hbox{%
952 \def\result{%
953 \kvsetknownkeys{FA}\cmd{key1=234,key3=456}%
954 \def\expected{[FA:key1=234]}%
955 \ifx\expected\result
956 \else
957 \Error{%
958 \string\kvsetknownkeys/\string\result\MessageBreak
959 Expected: \expected\MessageBreak

```

```

960      Result: \space\result
961    }%
962  \fi
963  \def\expected{{key3}={456}}%
964  \ifx\cmd\expected
965  \else
966    \Error{%
967      \string\kvsetknownkeys/\string\cmd\MessageBreak
968      Expected: \expected\MessageBreak
969      Result: \space\cmd
970    }%
971  \fi
972 }
973 \ifdim\wd0=0pt %
974 \else
975   \Error{Spurious spaces?}%
976 \fi

977 \csname @@end\endcsname\end
978 </test4>

```

## 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.

**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<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
```

---

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

**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
test/kvsetkeys-test4.tex → doc/latex/oberdiek/test/kvsetkeys-test4.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`.

## 5.4 Refresh file name databases

If your  $\text{\TeX}$  distribution (`te $\text{\TeX}$` , `mik $\text{\TeX}$` , ...) relies on file name databases, you must refresh these. For example, `te $\text{\TeX}$`  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  $\text{\LaTeX}$ .** The `.dtx` chooses its action depending on the format:

**plain  $\text{\TeX}$ :** Run `docstrip` and extract the files.

**$\text{\LaTeX}$ :** Generate the documentation.

If you insist on using  $\text{\LaTeX}$  for `docstrip` (really, `docstrip` does not need  $\text{\LaTeX}$ ), 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 $\text{\LaTeX}$` :

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

## 6 Catalogue

The following XML file can be used as source for the [T<sub>E</sub>X Catalogue](#). The elements `caption` and `description` are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is `kvsetkeys.xml`.

```
979 <*catalogue>
980 <?xml version='1.0' encoding='us-ascii'?>
981 <!DOCTYPE entry SYSTEM 'catalogue.dtd'>
982 <entry datestamp='$Date$' modifier='$Author$' id='kvsetkeys'>
983   <name>kvsetkeys</name>
984   <caption>Key value parser with default handler support.</caption>
985   <authorref id='auth:oberdiek'>
986     <copyright owner='Heiko Oberdiek' year='2006,2007,2009-2012'>
987       <license type='lppl1.3'>
988         <version number='1.16'>
989           <description>
990             This package provides \kvsetkeys, a variant of package
991             <xref refid='keyval'>keyval</xref>'s <tt>\setkeys</tt>. It allows
992             the user to specify a handler that deals with unknown options.
993             Active commas and equal signs may be used (e.g. see
994             <xref refid='babel'>babel</xref>'s shorthands) and only one level
995             of curly braces are removed from the values.
996           <p/>
997             The package is part of the <xref refid='oberdiek'>oberdiek</xref> bundle.
998         </description>
999         <documentation details='Package documentation'
1000           href='ctan:/macros/latex/contrib/oberdiek/kvsetkeys.pdf'>
1001           <ctan file='true' path='/macros/latex/contrib/oberdiek/kvsetkeys.dtx'>
1002             <miktex location='oberdiek'>
1003               <texlive location='oberdiek'>
1004                 <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'>
1005             </entry>
1006 </catalogue>
```

## 7 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](#).

## 8 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.
- $\varepsilon$ -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.

**[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  $\varepsilon$ -TeX.

## 9 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> .....	589, 931
<code>\\$</code> .....	189, 192
<code>\%</code> .....	665
<code>\&amp;</code> .....	190, 193
<code>\,</code> .....	236, 237, 740, 742, 749, 753, 756, 760, 810, 812
<code>\=</code> .....	36, 285, 286, 743, 745, 750, 754, 758, 762, 813
<code>\&gt;</code> .....	38, 39, 40, 41, 42, 43
<code>\@</code> .....	590, 663, 932
<code>\@PackageError</code> .....	529, 937
<code>\@Test</code> .....	755, 757, 759, 761, 764, 807
<code>\@UnDeFiNeD</code> .....	520
<code>\@ehc</code> .....	535, 937
<code>\@empty</code> .....	13, 854, 863, 895, 904
<code>\@endslash</code> .....	13, 16, 30
<code>\@firstofone</code> .....	598, 601
<code>\@gobble</code> .....	595, 603
<code>\@makeactive</code> .....	737, 740, 743, 756, 758, 810, 813
<code>\@makeoother</code> .....	742, 745, 749, 750, 753, 754, 760, 762, 812
<code>\@onelevel@sanitize</code> .....	799, 800, 851, 859, 900
<code>\@undefined</code> .....	105
<code>\@</code> ...	37, 38, 39, 40, 41, 42, 43, 44, 664
<code>\{</code> .....	587, 929
<code>\}</code> .....	588, 930
<code>\~</code> .....	237, 286
A	
<code>\advance</code> ..	551, 561, 628, 636, 651, 718
<code>\aftergroup</code> .....	76
<code>\AtEndDocument</code> .....	731
B	
<code>\begin</code> .....	34, 35, 747, 841, 891, 925
<code>\body</code> .....	607, 611
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, 189, 190, 192, 193, 587, 588, 589, 590, 625, 634, 642, 646, 663, 664, 665, 738, 929, 930, 931, 932
<code>\chardef</code> .....	692
<code>\cmd</code> .....	953, 964, 967, 969
<code>\comma@break</code> .....	6, 396
<code>\comma@entry</code> .....	387, 893
<code>\comma@list</code> .....	233, 379
<code>\comma@normalize</code> .....	6, 224, 378
<code>\comma@parse</code> .....	6, 377, 905
<code>\comma@parse@normalized</code> ..	6, 379, 381
<code>\count</code> .....	550, 551, 553, 560, 561, 563
<code>\count@</code> .....	592, 621, 625, 627, 628, 632, 634, 635, 636, 640, 642, 645, 646, 650, 651
<code>\countdef</code> .....	592
<code>\csname</code> .....	61, 68, 97, 113, 123, 163, 166, 176, 181, 183, 408, 419, 422, 426, 437, 440, 444, 450, 461, 464, 468, 479, 482, 486, 510, 513, 517, 520, 570, 582, 591, 594, 597, 600, 655, 682, 700, 977
D	
<code>\define@key</code> .....	29
<code>\dimexpr</code> .....	708
<code>\documentclass</code> .....	2, 688
<code>\dots</code> .....	38, 41, 43

<b>E</b>	
\empty	64, 65
\end	45, 46, 683, 839, 889, 923, 926, 977
\endcsname	61, 68, 97, 113, 123, 163, 166, 176, 181, 183, 408, 417, 419, 420, 422, 426, 435, 437, 440, 444, 450, 459, 461, 462, 464, 468, 477, 479, 482, 486, 510, 513, 517, 520, 570, 582, 591, 594, 597, 600, 655, 682, 700, 977
\endinput	76, 161
\endlinechar	51, 82, 118, 124, 136
\endqstest	722, 727
\errmessage	644, 741, 744
\Error	936, 957, 966, 975
\etex@unexpanded	199, 362, 494, 496
\Expect	766, 769, 843, 847, 865, 867, 893, 906, 908
\expected	954, 955, 959, 963, 964, 968
<b>H</b>	
\hbox	951
<b>I</b>	
\ifcase	180
\ifcsname	417, 420, 435, 459, 462, 477
\ifdim	973
\ifetex@unexpanded	180
\iffalse	400, 837, 838, 885, 886, 918, 919
\ifincsn	409, 451, 511, 518
\ifkv@handled@	400, 403, 405, 428, 470
\ifnum	627, 635, 642, 650
\iftrue	403, 409, 451, 511, 518
\ifx	62, 65, 68, 97, 105, 108, 163, 166, 176, 181, 183, 204, 294, 419, 422, 434, 437, 461, 464, 476, 479, 492, 499, 591, 594, 597, 600, 655, 700, 769, 846, 854, 895, 955, 964
\immediate	70, 99
\IncludeTests	697
\input	167, 656, 933, 934, 935
\iterate	608, 610, 612
<b>K</b>	
\kill	36
\kv@break	4, 374
\kv@define@key	940
\kv@error@generic	523, 526, 528
\kv@error@novalue	438, 480, 522
\kv@error@unknownkey	423, 430, 525
\kv@fam	416, 458
\kv@handled@false	399
\kv@handled@true	402, 405, 425, 467
\kv@key	353, 843
\kv@list	222, 335, 768, 769, 799
\kv@normalize	3, 210, 334, 765, 796, 797
\kv@parse	3, 333, 539, 542, 575, 864
\kv@parse@normalized	4, 335, 337
\kv@processor@default	4, 406, 539, 543
\kv@processor@known	4, 448, 575
\kv@set@family@handler	5, 20, 508
\kv@unset@family@handler	5, 515
\kv@value	355, 362, 367, 434, 476, 846
\KVS@@Comma	241, 243, 247
\KVS@@CommaComma	275, 277
\KVS@@CommaSpace	265, 267
\KVS@@Equals	290, 292, 306
\KVS@@EqualsSpace	324, 326
\KVS@@Process	358, 361
\KVS@@SpaceComma	253, 257
\KVS@@SpaceEquals	312, 316
\KVS@AddUnhandled	465, 472, 490
\KVS@AtEnd	142, 143, 161, 584
\KVS@break	371, 375, 397
\KVS@cmd	546, 548, 550, 553, 558, 560, 563, 570, 582
\KVS@cmd@dec	557, 576
\KVS@cmd@inc	547, 574
\KVS@Comma	213, 227, 235
\KVS@CommaComma	216, 230, 274
\KVS@CommaParse	382, 384
\KVS@CommaSpace	215, 229, 264
\KVS@Empty	195, 204, 294
\KVS@empty	492, 499, 567, 573
\KVS@Equals	217, 284
\KVS@EqualsSpace	219, 323
\KVS@FirstOfTwo	196, 205, 295
\KVS@Global	220, 222, 231, 233, 365, 367, 504, 506
\KVS@IfEmpty	198, 245, 258, 268, 278, 304, 317, 327, 341, 346, 354, 385, 391
\KVS@MaybeBreak	345, 352, 370, 372, 375, 390, 397
\KVS@Nil	241, 243, 247, 253, 257, 261, 265, 267, 271, 275, 277, 281, 290, 292, 306, 312, 316, 320, 324, 326, 330, 338, 340, 343, 348, 351, 358, 361, 382, 384, 393
\KVS@Parse	338, 340
\KVS@Process	343, 351
\KVS@ProcessorDefault	411, 415
\KVS@ProcessorKnown	453, 457
\KVS@SecondOfTwo	197, 207, 297
\KVS@setknownkeys	570, 572, 581
\KVS@SpaceComma	214, 228, 251
\KVS@SpaceEquals	218, 310
\KVS@Temp	199, 202, 204, 293, 294
\KVS@temp	410, 413, 452, 455
\KVS@TestMode	692
\kvsetkeys	5, 14, 538, 990
\kvsetkeys@expandafter	6, 541
\kvsetknownkeys	5, 568, 953, 958, 967
\kvsetknownkeys@expandafter	579
<b>L</b>	
\lccode	236, 237, 285, 286
\LoadCommand	656, 666
\LogTests	698
\loop	606, 622, 633, 641
\lowercase	238, 287
<b>M</b>	
\makeatletter	7, 691, 702, 736

