Abstract

Package \texttt{kvsetkeys} provides \texttt{\kvsetkeys}, a variant of package keyval’s \texttt{\setkeys}. It allows to specify a handler that deals with unknown options. Active commas and equal signs may be used (e.g. see \texttt{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 \texttt{keyval} ................. 4
    1.4.2 Processing similar to \texttt{\setkeys*} of package xkeyval .... 4
  1.5 Default family handler ................................ 5
  1.6 Put it all together ..................................... 5
  1.7 Comma separated lists .................................. 6

2 Example ..................................................................... 6

3 Implementation .................................................... 7
  3.1 Identification .............................................. 7
  3.2 Package loading ........................................... 9
  3.3 Check for \texttt{\epsilon-TEX} .............................. 9
  3.4 Generic help macros ..................................... 10
  3.5 Normalizing ................................................. 10
  3.6 Parsing key value lists .................................. 12
  3.7 Parsing comma lists ..................................... 14
  3.8 Processing key value pairs ............................ 14
  3.9 Error handling ............................................ 17
  3.10 Do it all .................................................... 17

4 Installation ........................................................ 18
  4.1 Download ................................................... 18
  4.2 Package installation ..................................... 18
  4.3 Refresh file name databases ............................ 19
  4.4 Some details for the interested ....................... 19

5 References .......................................................... 19

*Please report any issues at \url{https://github.com/ho-tex/kvsetkeys/issues}
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

\texttt{kvsetkeys} serves as replacement for \texttt{keyval}'s \texttt{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 \texttt{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 \texttt{babel} uses the equals sign as active shorthand character. Therefore package \texttt{kvsetkeys} deals with both catcode settings 12 (other) and 13 (active).

**Brace removal:** Package \texttt{keyval}'s \texttt{setkeys} removes up to two levels of curly braces around the value in some unpredictable way:

\begin{verbatim}
\setkeys{fam}{key={value}} \rightarrow value
\end{verbatim}
\begin{verbatim}
\setkeys{fam}{key={value}} \rightarrow {value}
\end{verbatim}
\begin{verbatim}
\setkeys{fam}{key= {{value}}} \rightarrow {value}
\end{verbatim}

This package \texttt{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.

\begin{verbatim}
\kvsetkeys{fam}{key={value}} \rightarrow value
\end{verbatim}
\begin{verbatim}
\kvsetkeys{fam}{key={value}} \rightarrow {value}
\end{verbatim}
\begin{verbatim}
\kvsetkeys{fam}{key= {{value}}} \rightarrow {value}
\end{verbatim}
Arbitrary values: Unmatched conditionals are supported.

Before I describe \texttt{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

\begin{verbatim}
\texttt{\kv@normalize}\{⟨key value list⟩\}
\end{verbatim}

If the user specifies key value lists, he usually prefers nice formatted source code, e.g.:

\begin{verbatim}
\hypersetup{
pdftitle = {...},
pdfssubject = {...},
pdfauthor = {...},
pdfkeywords = {...},
...}
\end{verbatim}

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 \texttt{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.

\texttt{\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: \texttt{babel}'s language option \texttt{turkish} uses the equal sign as active shorthand character.)

The result is stored in \texttt{\kv@list}, e.g.:

\begin{verbatim}
\kv@list → ,pdftitle={...},pdfsubject={...},...
\end{verbatim}

Curly braces around values (or keys) remain untouched.

\texttt{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.

\texttt{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

\begin{verbatim}
\texttt{\kv@parse}\{⟨key value list⟩\}\{⟨processor⟩\}
\end{verbatim}

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

\begin{verbatim}
\texttt{\kv@parse@normalized}\{⟨key value list⟩\}\{⟨processor⟩\}
\end{verbatim}

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 \texttt{⟨processor⟩}:

\texttt{⟨processor⟩}\{⟨key⟩\}\{⟨value⟩\}
Also key and value are stored in macro names:

- \texttt{\textbackslash kv\textbackslash key} stores the key.
- \texttt{\textbackslash kv\textbackslash value} stores the value or if the value was not specified it has the meaning \texttt{\textbackslash relax}.

The behaviour in pseudo code:

\begin{verbatim}
foreach (⟨key⟩, ⟨value⟩) in ((key value list))
  \texttt{\textbackslash kv\textbackslash key := ⟨key⟩}
  \texttt{\textbackslash kv\textbackslash value := ⟨value⟩}
  \texttt{(processor) {⟨key⟩} {⟨value⟩}}
\end{verbatim}

Since version 2011/03/03 v1.11 \texttt{\textbackslash kv\textbackslash break} can be called inside the \texttt{(processor)} of \texttt{\textbackslash kv\textbackslash parse} or \texttt{\textbackslash kv\textbackslash 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 \texttt{\textbackslash kvsetkeys} works similar to \texttt{\textbackslash setkeys} of package keyval. There unknown keys raise an error.

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

1.4.1 Processing similar to keyval

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

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

Since 2012/04/25 v1.16 \texttt{\textbackslash kv\textbackslash processor@default} also defines macro \texttt{\textbackslash kv\textbackslash fam} with meaning \texttt{⟨family⟩} for convenience.

1.4.2 Processing similar to \texttt{\textbackslash setkeys\textast}} of package xkeyval

\texttt{\textbackslash kv\textbackslash processor@known {⟨family⟩} {⟨cmd⟩} {⟨key⟩} {⟨value⟩}}

The key value processor \texttt{\textbackslash kv\textbackslash processor@known} behaves similar to \texttt{\textbackslash kv\textbackslash processor@default}. If the \texttt{⟨key⟩} exists in the \texttt{⟨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 \texttt{⟨cmd⟩}. Since 2011/10/18 v1.15.

The behaviour in pseudo code:

\begin{verbatim}
if ⟨key⟩ exists
  call the keyval code of ⟨key⟩
else
  if ⟨handler⟩ for ⟨family⟩ exists
\end{verbatim}
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

Since 2012/04/25 v1.16 \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⟩
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. \texttt{(processor)} is called for each non-empty entry with the entry as argument:

\texttt{(processor)}\{\texttt{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 \texttt{(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.

\begin{verbatim}
1 ⟨∗example⟩
2 \documentclass{article}
3 \usepackage[T1]{fontenc}
4 \usepackage{kvsetkeys}
5 \usepackage{keyval}
6 \makeatletter
7 \newcommand*{⟨tag⟩}[2][]{%
8 % #1: attributes
\end{verbatim}
3 Implementation

3.1 Identification

Reload check, especially if the package is not used with LaTeX.
\ifx\empty % LaTeX, first loading, 
% variable is initialized, but \ProvidesPackage not yet seen 
\else
\expandafter\ifx\csname PackageInfo\endcsname\relax
\def\x#1#2{% 
\immediate\write-1{Package #1 Info: #2.}%
}\)
\else
\fi
\x{kvsetkeys}{The package is already loaded}%
\aftergroup\endinput
\fi
\fi
\endgroup

Package identification:
\begin{group}
\catcode61\catcode48\catcode32=10\relax\n\catcode13=5 % ^^M
\endlinechar=13 %
\catcode35=6 % #
\catcode39=12 % '
\catcode40=12 % ( 
\catcode41=12 % )
\catcode44=12 % ,
\catcode45=12 % -
\catcode46=12 % .
\catcode47=12 % /
\catcode58=12 % :
\catcode64=11 % @
\def\x{\endgroup
\expandafter\edef\csname KVS@AtEnd\endcsname{\endlinechar=\the\endlinechar\relax
\catcode13=\the\catcode13\relax
\endgroup}
\expandafter\ifx\csname ver@kvsetkeys.sty\endcsname
\ProvidesPackage{kvsetkeys}[
2022-10-05 v1.19 Key value parser (HO)]
\begingroup\catcode61\catcode48\catcode32=10\relax\n\catcode13=5 % ^^M
\endlinechar=13 %
\catcode32=10 %
\edef\x{\endgroup
\expandafter\edef\csname KVS@AtEnd\endcsname{\endlinechar=\the\endlinechar\relax
\catcode13=\the\catcode13\relax
}}}
3.2 Package loading

\begin{Verbatim}
\begin{group}
\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
\def\TMP@RequirePackage#1[#2]{\begingroup
\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname ver@#1.sty\endcsname\relax
\input #1.sty\relax
\else\fi
\expandafter\ifx\csname toks@\endcsname\relax
\toksdef\toks@=0\fi
\endgroup
\end{Verbatim}

3.3 Check for \textit{e-T\TeX\xspace}

\unexpanded, \texttt{\ifcsname}, and \texttt{\unless} are used if found.
\begin{Verbatim}
\begin{group}
\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
\def\TMP@RequirePackage#1[#2]{\begingroup
\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname ver@#1.sty\endcsname\relax
\input #1.sty\relax
\else\fi
\expandafter\ifx\csname toks@\endcsname\relax
\toksdef\toks@=0\fi
\endgroup
\end{Verbatim}

\begin{Verbatim}
\begin{group}
\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
\def\TMP@RequirePackage#1[#2]{\begingroup
\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname ver@#1.sty\endcsname\relax
\input #1.sty\relax
\else\fi
\expandafter\ifx\csname toks@\endcsname\relax
\toksdef\toks@=0\fi
\endgroup
\end{Verbatim}
\texttt{\catcode`\&=9 % ignore}
\texttt{\fi}

### 3.4 Generic help macros

\texttt{\KVS@Empty}
\begin{verbatim}
\def\KVS@Empty{}
\end{verbatim}

\texttt{\KVS@FirstOfTwo}
\begin{verbatim}
\long\def\KVS@FirstOfTwo#1#2{#1}
\end{verbatim}

\texttt{\KVS@SecondOfTwo}
\begin{verbatim}
\long\def\KVS@SecondOfTwo#1#2{#2}
\end{verbatim}

\texttt{\KVS@IfEmpty}
\begin{verbatim}
\long\def\KVS@IfEmpty#1{\begingroup&\edef\KVS@Temp{\unexpanded{#1}}\begingroup$\toks@{#1}$\edef\KVS@Temp{\the	oks@}$\expandafter\endgroup\ifx\KVS@Temp\KVS@Empty\expandafter\KVS@FirstOfTwo\else\expandafter\KVS@SecondOfTwo\fi\fi}\end{verbatim}

### 3.5 Normalizing

\texttt{\kv@normalize}
\begin{verbatim}
\long\def\kv@normalize#1{\begingroup\toks@{,#1,}\KVS@Comma\KVS@SpaceComma\KVS@CommaSpace\KVS@CommaComma\KVS@Equals\KVS@SpaceEquals\KVS@EqualsSpace\xdef\KVS@Global{\the	oks@}\endgroup\let\kv@list\KVS@Global}
\end{verbatim}

\texttt{\comma@normalize}
\begin{verbatim}
\def\comma@normalize#1{\begingroup\toks@{,#1,}\KVS@Comma\KVS@SpaceComma\KVS@CommaSpace\KVS@CommaComma\xdef\KVS@Global{\the\toks@}\endgroup\let\comma@list\KVS@Global}
\end{verbatim}

\texttt{\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.
\begingroup
\lccode\,='\%
\lccode\~='\%
\lowercase{endgroup}
\def\KVS@Comma{%
\toks0\expandafter{\expandafter}\expandafter\KVS@@Comma\the\toks0~\KVS@Nil
}
\long\def\KVS@@Comma#1~#2\KVS@Nil{%
\toks0\expandafter{\the\toks0#1}%
\KVS@IfEmpty{#2}{%}
\KVS@Comma,#2\KVS@Nil
}
\KVS@SpaceComma
Removes spaces before the comma, may add commas at the end.
\def\KVS@SpaceComma{%
\def\KVS@SpaceComma{%
\expandafter\KVS@@SpaceComma\the\toks0#1,\KVS@Nil
}
\KVS@@SpaceComma
long\def\KVS@@SpaceComma#1, #2\KVS@Nil{%
\KVS@IfEmpty{#2}{%}
\toks0{#1}%
\KVS@@SpaceComma#1,#2\KVS@Nil
}
\KVS@CommaSpace
Removes spaces after the comma, may add commas at the end.
\def\KVS@CommaSpace{%
\expandafter\KVS@@CommaSpace\the\toks0, \KVS@Nil
}
\KVS@@CommaSpace
long\def\KVS@@CommaSpace#1, #2\KVS@Nil{%
\KVS@IfEmpty{#2}{%}
\toks0(#1)%
\KVS@CommaSpace#1,#2\KVS@Nil
}
\KVS@CommaComma
Replaces multiple commas by one comma.
\def\KVS@CommaComma{%
\expandafter\KVS@@CommaComma\the\toks0, \KVS@Nil
}
\KVS@@CommaComma
long\def\KVS@@CommaComma#1,, #2\KVS@Nil{%
\KVS@IfEmpty{#2}{%}
\toks0(#1)% ()
\KVS@CommaComma#1,#2\KVS@Nil
}
\endgroup
\KVS@Equals Converts active equals signs into catcode other characters.
\begingroup
\lccode'\==\=%
\lccode'\~\==\%
\lowercase{\endgroup
\def\KVS@Equals{\toks@\expandafter{\expandafter}
\KVS@@Equals\the\toks@~\KVS@Nil}
\long\def\KVS@@Equals#1~#2\KVS@Nil{\edef\KVS@Temp{\the\toks@}\
\ifx\KVS@Temp\KVS@Empty
\expandafter\KVS@FirstOfTwo
\else
\expandafter\KVS@SecondOfTwo
\fi
\toks@{#1}\
\toks@{#1=#2}\KVS@Nil}
\KVS@IfEmpty{#2}{\KVS@Nil}
}

\KVS@SpaceEquals Removes spaces before the equals sign.
\def\KVS@SpaceEquals#1{\def\KVS@SpaceEquals{\
\expandafter\KVS@@SpaceEquals\the\toks@#1=\KVS@Nil}
\KVS@@SpaceEquals#1 =#2\KVS@Nil{\KVS@IfEmpty{#2}{\toks@{#1}\
\KVS@IfEmpty{#2}{\KVS@Nil}
}
}

\KVS@EqualsSpace Removes spaces after the equals sign.
\def\KVS@EqualsSpace{\expandafter\KVS@@EqualsSpace\the\toks@= \KVS@Nil
\KVS@@EqualsSpace#1= #2\KVS@Nil{\KVS@IfEmpty{#2}{\toks@{#1}\
\KVS@IfEmpty{#2}{\KVS@Nil}
}
}

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 & \edef\kv@value{\unexpanded{#2}}%
342 \begingroup
343 $ \toks@{#2}$
344 \xdef\KVS@Global{\the\toks@}$
345 $ \let\kv@value\KVS@Global$
346 #3{#1}{#2}%
347 \endgroup
348 \}
349 \}
\KVS@MaybeBreak
350 \let\KVS@MaybeBreak\relax
\KVS@break
351 \long\def\KVS@break#1#2#3#4{%
352 \let\KVS@MaybeBreak\relax
353 \let\KVS@MaybeBreak\relax
354 \def\KVS@Global{\the\toks@}$
355 $ \edef\kv@value\KVS@Global$
356 #3{#1}{#2}%
357 \}
358 \}
\KVS@break
359 \let\KVS@MaybeBreak\relax
3.7 Parsing comma lists

\comma@parse Normalizes and parses the key value list. Also sets \comma@list.
\def\comma@parse#1{%  
  \comma@normalize{#1}%  
  \expandafter\comma@parse@normalized\expandafter{\comma@list}%}  
\comma@parse@normalized #1: comma list  
#2: processor
\def\comma@parse@normalized#1#2{%  
  \KVS@CommaParse#1,#2\KVS@Nil#3{%  
    \KVS@IfEmpty{#1}{%  
    }{%  
      \def\comma@entry{#1}%  
      #3{#1}%  
    }%  
  }%  
  \KVS@MaybeBreak  
  \KVS@IfEmpty{#2}{%  
    }{%  
      \KVS@CommaParse#2\KVS@Nil{#3}%  
    }%  
}  
\comma@break
\def\comma@break{%  
  \let\KVS@MaybeBreak\KVS@break%}  

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).
\def\kv@handled@false{%  
  \let\ifkv@handled@iffalse%}  
\kv@handled@true
\def\kv@handled@true{%  
  \let\ifkv@handled@iftrue%}  
\ifkv@handled@
\kv@handled@true
\def\kv@handled@default{%  
  \begin{verbatim}  
  \begin{verbatim}  
  \end{verbatim}  
  \end{verbatim}  
  \classname \@safe@activetrue\endclassname
}
\KVS@AddUnhandled\#2{\#3}{\#4}\% 
\else 
\kv@handled@true 
\csname KV@\#1@handler\endcsname{\#3}{\#4}\% 
\relax
\ifkv@handled@ 
\else 
\KVS@AddUnhandled\#2{\#3}{\#4}\% 
\fi 
\fi 
\else 
\ifx\kv@value\relax & \unless\ifcsname KV@\#1@\#2@default\endcsname 
$ \begingroup\expandafter\expandafter\expandafter\endgroup $ \expandafter\ifx\csname KV@\#1@\#3@default\endcsname\relax 
\kv@error@novalue{\#1}{\#3} \% 
\else 
\csname KV@\#1@\#3@default\endcsname \relax 
\fi 
\else 
\csname KV@\#1@\#3\endcsname{\#4} \% 
\fi 
\fi 
} 
\KVS@AddUnhandled\long\def\KVS@AddUnhandled\#1\#2\#3\% \edef\#1\% \if\#1\KVS@empty \else \unexpanded\expandafter\{\#1\},\% \fi \unexpanded\{\#2=}\{\#3\}\% \} \begingroup \ifaux\KVS@empty \toks@\{\#2=}\{\#3\}% \else \toks@\expandafter\{\#1,\#2=}\{\#3\}% \fi \edef\KVS@Global{\the\toks@} \let\#1\KVS@Global \} 
\kv@set@family@handler 
\long\def\kv@set@family@handler\#1\#2\% \begingroup \csname @safe@activestrue\endcsname \let\ifincsname\iftrue \expandafter\endgroup \expandafter\def\csname KVS@\#1@handler\endcsname##1##2{#2} \}% 
\kv@unset@family@handler \long\def\kv@unset@family@handler\#1\#2\% \begingroup \csname @safe@activestrue\endcsname \let\ifincsname\iftrue \expandafter\endgroup \expandafter\let\csname KVS@\#1@handler\endcsname\@UnDeFiNeD
3.9 Error handling

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

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

\kv@error@generic
517 \def\kv@error@generic#1#2#3{\PackageError{kvsetkeys}{#1 key ‘#3’}{The keyval family of the key ‘#3’ is ‘#2’.\MessageBreakThe setting of the key is ignored because of the error.\MessageBreak\@ehc}}
518 }

3.10 Do it all

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

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

\KVS@cmd
535 \def\KVS@cmd{0}

\KVS@cmd@inc
536 \def\KVS@cmd@inc{% \edef\KVS@cmd\the\numexpr\KVS@cmd+1\relax
538 $ \begingroup \count255=\KVS@cmd\relax
540 $ \advance\count255 by 1\relax
541 $ \edef\x{\endgroup\def\KVS@cmd{\number\count255}}$
543 $ \x$
545 }

\KVS@cmd@dec
546 \def\KVS@cmd@dec{% \edef\KVS@cmd\the\numexpr\KVS@cmd-1\relax
548 $ \begingroup \count255=\KVS@cmd\relax
550 $ \advance\count255 by -1\relax
551 $ \edef\x{\endgroup\def\KVS@cmd{\number\count255}}$
553 $ \x$
555 }
4 Installation

4.1 Download

Package. This package is available on CTAN:\footnote{CTAN:pkg/kvsetkeys}

CTAN:macros/latex/contrib/kvsetkeys/kvsetkeys.dtx The source file.

4.2 Package installation

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

\texttt{tex kvsetkeys.dtx}

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

\begin{itemize}
  \item \texttt{kvsetkeys.sty} \rightarrow \texttt{tex/generic/kvsetkeys/kvsetkeys.sty}
  \item \texttt{kvsetkeys.pdf} \rightarrow \texttt{doc/latex/kvsetkeys/kvsetkeys.pdf}
  \item \texttt{kvsetkeys-example.tex} \rightarrow \texttt{doc/latex/kvsetkeys/kvsetkeys-example.tex}
  \item \texttt{kvsetkeys.dtx} \rightarrow \texttt{source/latex/kvsetkeys/kvsetkeys.dtx}
\end{itemize}

If you have a \texttt{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.
4.3 Refresh file name databases
If your TeX distribution (TeX Live, MiKTeX, ...) relies on file name databases, you must refresh these. For example, TeX Live users run `texhash` or `mktexlsr`.

4.4 Some details for the interested
Unpacking with LATEX. The `.dtx` chooses its action depending on the format:

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

\LATEX{}: Generate the documentation.

If you insist on using \LaTeX{} for `docstrip` (really, `docstrip` does not need \LaTeX{}), then inform the autodetect routine about your intention:

\begin{verbatim}
latex \let\install=y\input{kvsetkeys.dtx}
\end{verbatim}

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:

\begin{verbatim}
\PassOptionsToClass{a4paper}{article}
\end{verbatim}

An example follows how to generate the documentation with pdflatex:

\begin{verbatim}
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
\end{verbatim}

5 References


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.
- $\varepsilon$-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, \ldots, \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 infwarerr in LATEX.

[2022-10-05 v1.19]
- Corrected storing of unknown keys, issue #1

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

\$ ................................. 178, 181
\& ................................. 179, 182
\, ................................. 225, 226
\= ................................. 36, 274, 275
\> ................................. 38, 39, 40, 41, 42, 43
\@PackageError ................................. 171
\@UnDeFiNeD ................................. 509
@ehc ................................ 524
@empty ................................. 13
@endslash ................................. 13, 16, 30
\@undefined ................................. 105, 177
\\ ................................. 37, 38, 39, 40, 41, 42, 43, 44
\` ................................ 226, 275
\advance ................................ 540, 550
\aftergroup ................................. 76
\begin ................................ 34, 35
\catcode 49, 50, 52, 53, 54, 55, 56,

A 57, 58, 59, 60, 80, 81, 83, 84, 85,
B 86, 87, 88, 89, 90, 91, 92, 93, 94,
C 95, 96, 116, 117, 119, 120, 121,
D 125, 126, 127, 128, 129, 130,
E 131, 134, 135, 137, 138, 139,
\count ................................ 140, 144, 146, 178, 179, 181, 182
\comma@break ................................. 6, 385
\comma@entry ................................. 376
\comma@list ................................. 222, 368
\comma@normalize ................................. 6, 213, 367
\comma@parse ................................. 6, 366
\comma@parse@normalized ................................. 6, 368, 370
\count ................................ 539, 540, 542, 549, 550, 552
\csname ................................ 61, 68,
\advance ................................ 97, 113, 123, 163, 166, 174, 397,
\aftergroup ................................. 408, 411, 415, 426, 429, 433,
\begin ................................ 439, 450, 453, 457, 468, 471,
\catcode ................................. 475, 499, 502, 506, 509, 559, 571
\define@key ................................. 29
\documentclass ................................. 2
\dots ................................ 38, 41, 43