The **kvsetkeys** package

Heiko Oberdiek*

2022-10-05 v1.19

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 `\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 ε-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 https://github.com/ho-tex/kvsetkeys/issues

6 History 19
[2006/03/06 v1.0] 19
[2006/10/19 v1.1] 19
[2007/09/09 v1.2] 19
[2007/09/29 v1.3] 20
[2009/07/19 v1.4] 20
[2009/07/30 v1.5] 20
[2009/12/12 v1.6] 20
[2009/12/22 v1.7] 20
[2010/01/28 v1.8] 20
[2010/03/01 v1.9] 20
[2011/01/30 v1.10] 20
[2011/03/03 v1.11] 20
[2011/04/05 v1.12] 20
[2011/04/07 v1.13] 20
[2011/06/15 v1.14] 20
[2011/10/18 v1.15] 21
[2012/04/25 v1.16] 21
[2016/05/16 v1.17] 21
[2019/12/15 v1.18] 21
[2022-10-05 v1.19] 21

7 Index 21

1 Documentation

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

1.1 Motivation

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

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

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

```
\setkeys{fam}{key={{value}}}      → 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` {⟨*key value list*⟩} {⟨*processor*⟩}

Now the key value list is split into single key value pairs. For further processing the key and value are given as arguments for the ⟨*processor*⟩:

⟨*processor*⟩ {⟨*key*⟩} {⟨*value*⟩}

Also key and value are stored in macro names:

- `\kv@key` stores the key.
- `\kv@value` stores the value or if the value was not specified it has the meaning `\relax`.

The behaviour in pseudo code:

```
foreach (⟨key⟩, ⟨value⟩) in (⟨key value list⟩)
    \kv@key := ⟨key⟩
    \kv@value := ⟨value⟩
    ⟨processor⟩ {⟨key⟩} {⟨value⟩}
```

`\kv@break`

Since version 2011/03/03 v1.11 `\kv@break` can be called inside the ⟨*processor*⟩ 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` {⟨*family*⟩} {⟨*key*⟩} {⟨*value*⟩}

There are many possiblities 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 processer, however, calls a handler instead, if it is provided by the family. Both ⟨*family*⟩ and ⟨*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 `\kv@handled@false`.

Since 2012/04/25 v1.16 `\kv@processor@default` also defines macro `\kv@fam` with meaning ⟨*family*⟩ 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 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@hander`:

`\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`.

`\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.

```
⟨*example⟩
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{kvsetkeys}
\usepackage{keyval}

\makeatletter
\newcommand*{\tag}[2][]{%
  % #1: attributes
  % #2: tag name
  \begingroup
    \toks@={}%
    \let\@endslash\@empty
    \kvsetkeys{tag}{#1}%
    \texttt{%
      \textless #2\the\toks@\@endslash\textgreater
    }%
  \endgroup
}
\kv@set@family@handler{tag}{%
  % #1: key
  % #2: value
  \toks@\expandafter{%
    \the\toks@
    \space
    #1=\string"#2\string"%
  }%
}
\define@key{tag}{/}[]{%
  \def\@endslash{/}%
}
\makeatother

\begin{document}
\begin{tabbing}
  \mbox{}\qquad\=\qquad\=\kill
  \tag{html}\\
  \>\dots\\
  \>\tag[border=1]{table}\\
  \>\>\tag[width=200, span=3, /]{colgroup}\\
  \>\>\dots\\
  \>\tag{/table}\\
  \>\dots\\
  \tag{/html}\\
\end{tabbing}
\end{document}
⟨/example⟩
```

3 Implementation

3.1 Identification

```
⟨*package⟩
```

Reload check, especially if the package is not used with LaTeX.

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode35=6 % #
  \catcode39=12 % '
  \catcode44=12 % ,
  \catcode45=12 % -
  \catcode46=12 % .
  \catcode58=12 % :
  \catcode64=11 % @
  \catcode123=1 % {
  \catcode125=2 % }
  \expandafter\let\expandafter\x\csname ver@kvsetkeys.sty\endcsname
  \ifx\x\relax % plain-TeX, first loading
  \else
    \def\empty{}%
    \ifx\x\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
        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
      \fi
      \x{kvsetkeys}{The package is already loaded}%
      \aftergroup\endinput
    \fi
  \fi
\endgroup%
```

Package identification:

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \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 % @
  \catcode91=12 % [
  \catcode93=12 % ]
  \catcode123=1 % {
  \catcode125=2 % }
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\@undefined
        \xdef#1{#3}%
      \fi
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@kvsetkeys.sty\endcsname
\ProvidesPackage{kvsetkeys}%
  [2022-10-05 v1.19 Key value parser (HO)]%
```

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode123=1 % {
  \catcode125=2 % }
  \catcode64=11 % @
  \def\x{\endgroup
    \expandafter\edef\csname KVS@AtEnd\endcsname{%
      \endlinechar=\the\endlinechar\relax
      \catcode13=\the\catcode13\relax
      \catcode32=\the\catcode32\relax
      \catcode35=\the\catcode35\relax
      \catcode61=\the\catcode61\relax
      \catcode64=\the\catcode64\relax
      \catcode123=\the\catcode123\relax
      \catcode125=\the\catcode125\relax
    }%
  }%
\x\catcode61\catcode48\catcode32=10\relax%
\catcode13=5 % ^^M
\endlinechar=13 %
\catcode35=6 % #
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\KVS@AtEnd{%
    \KVS@AtEnd
    \catcode#1=\the\catcode#1\relax
  }%
  \catcode#1=#2\relax
}
\TMP@EnsureCode{36}{3}% $
\TMP@EnsureCode{38}{4}% &
\TMP@EnsureCode{39}{12}% '
\TMP@EnsureCode{43}{12}% +
\TMP@EnsureCode{44}{12}% ,
\TMP@EnsureCode{45}{12}% -
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{91}{12}% [
\TMP@EnsureCode{93}{12}% ]
\TMP@EnsureCode{94}{7}% ^ (superscript)
\TMP@EnsureCode{96}{12}% `
\TMP@EnsureCode{126}{13}% ~ (active)
\edef\KVS@AtEnd{\KVS@AtEnd\noexpand\endinput}
```

3.2 Package loading

```
\begingroup\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
    \fi
  }%
  \TMP@RequirePackage{infwarerr}[2007/09/09]%
  \let\PackageError\@PackageError
\else
\fi
```

```
\expandafter\ifx\csname toks@\endcsname\relax
  \toksdef\toks@=0 %
\fi
```

3.3 Check for ε-TeX

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

```
\ifx\numexpr\@undefined
  \catcode`\$=9 % ignore
  \catcode`\&=14 % comment
\else % e-TeX
  \catcode`\$=14 % comment
  \catcode`\&=9  % ignore
\fi
```

3.4 Generic help macros

\KVS@Empty

```
\def\KVS@Empty{}
```

\KVS@FirstOfTwo

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

\KVS@SecondOfTwo

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

\KVS@IfEmpty

```
\long\def\KVS@IfEmpty#1{%
& \edef\KVS@Temp{\unexpanded{#1}}%
$ \begingroup
$   \toks@{#1}%
$   \edef\KVS@Temp{\the\toks@}%
$ \expandafter\endgroup
  \ifx\KVS@Temp\KVS@Empty
    \expandafter\KVS@FirstOfTwo
  \else
    \expandafter\KVS@SecondOfTwo
  \fi
}
```

3.5 Normalizing

\kv@normalize

```
\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\toks@}%
  \endgroup
  \let\kv@list\KVS@Global
}
```

\comma@normalize

```
\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
}
```

\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{%
    \toks@\expandafter{\expandafter}\expandafter
    \KVS@@Comma\the\toks@~\KVS@Nil
  }%
  \long\def\KVS@@Comma#1~#2\KVS@Nil{%
    \toks@\expandafter{\the\toks@#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#1{%
  \def\KVS@SpaceComma{%
    \expandafter\KVS@@SpaceComma\the\toks@#1,\KVS@Nil
  }%
}
\KVS@SpaceComma{ }
```

`\KVS@@SpaceComma`

```
\long\def\KVS@@SpaceComma#1 ,#2\KVS@Nil{%
  \KVS@IfEmpty{#2}{%
    \toks@{#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\toks@, \KVS@Nil
}
```

`\KVS@@CommaSpace`

```
\long\def\KVS@@CommaSpace#1, #2\KVS@Nil{%
  \KVS@IfEmpty{#2}{%
    \toks@{#1}%
  }{%
    \KVS@@CommaSpace#1,#2\KVS@Nil
  }%
}
```

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

```
\def\KVS@CommaComma{%
  \expandafter\KVS@@CommaComma\the\toks@,\KVS@Nil
}
```

`\KVS@@CommaComma`

```
\long\def\KVS@@CommaComma#1,,#2\KVS@Nil{%
  \KVS@IfEmpty{#2}{%
    \toks@{#1,}% (!)
  }{%
    \KVS@@CommaComma#1,#2\KVS@Nil
  }%
}
```

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

```
\begingroup
  \lccode`\==`\=%
  \lccode`\~=`\=%
\lowercase{\endgroup
  \def\KVS@Equals{%
    \toks@\expandafter{\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@\expandafter{\the\toks@=#1}%
    }%
    \KVS@IfEmpty{#2}{%
    }{%
      \KVS@@Equals#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{ }
```

`\KVS@@SpaceEquals`

```
\long\def\KVS@@SpaceEquals#1 =#2\KVS@Nil{%
  \KVS@IfEmpty{#2}{%
    \toks@{#1}%
  }{%
    \KVS@@SpaceEquals#1=#2\KVS@Nil
  }%
}
```

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

```
\def\KVS@EqualsSpace{%
  \expandafter\KVS@@EqualsSpace\the\toks@= \KVS@Nil
}
```

`\KVS@@EqualsSpace`

```
\long\def\KVS@@EqualsSpace#1= #2\KVS@Nil{%
  \KVS@IfEmpty{#2}{%
    \toks@{#1}%
  }{%
    \KVS@@EqualsSpace#1=#2\KVS@Nil
  }%
}
```

3.6 Parsing key value lists

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

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

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

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

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

```
\long\def\KVS@Parse#1,#2\KVS@Nil#3{%
  \KVS@IfEmpty{#1}{%
  }{%
    \KVS@Process#1=\KVS@Nil{#3}%
  }%
  \KVS@MaybeBreak
  \KVS@IfEmpty{#2}{%
  }{%
    \KVS@Parse#2\KVS@Nil{#3}%
  }%
}
```

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

```
\long\def\KVS@Process#1=#2\KVS@Nil#3{%
  \let\KVS@MaybeBreak\relax
  \def\kv@key{#1}%
  \KVS@IfEmpty{#2}{%
    \let\kv@value\relax
    #3{#1}{}%
  }{%
    \KVS@@Process{#1}#2\KVS@Nil{#3}%
  }%
}
```

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

```
\long\def\KVS@@Process#1#2=\KVS@Nil#3{%
& \edef\kv@value{\unexpanded{#2}}%
$ \begingroup
$   \toks@{#2}%
$   \xdef\KVS@Global{\the\toks@}%
$ \endgroup
$ \let\kv@value\KVS@Global
  #3{#1}{#2}%
}
```

`\KVS@MaybeBreak`

```
\let\KVS@MaybeBreak\relax
```

`\KVS@break`

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

`\kv@break`

```
\def\kv@break{%
  \let\KVS@MaybeBreak\KVS@break
}
```

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,\KVS@Nil{#2}%
}
```

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

```
\def\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
```

`\kv@processor@default`

```
\def\kv@processor@default#1#2{%
  \begingroup
    \csname @safe@activestrue\endcsname
    \let\ifincsname\iftrue
    \edef\KVS@temp{\endgroup
      \noexpand\KVS@ProcessorDefault{#1}{#2}%
    }%
  \KVS@temp
}
```

\KVS@ProcessorDefault

```
\long\def\KVS@ProcessorDefault#1#2#3{%
  \def\kv@fam{#1}%
& \unless\ifcsname KV@#1@#2\endcsname
$ \begingroup\expandafter\expandafter\expandafter\endgroup
$ \expandafter\ifx\csname KV@#1@#2\endcsname\relax
&   \unless\ifcsname KVS@#1@handler\endcsname
$   \begingroup\expandafter\expandafter\expandafter\endgroup
$   \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
      \kv@error@unknownkey{#1}{#2}%
    \else
      \kv@handled@true
      \csname KVS@#1@handler\endcsname{#2}{#3}%
      \relax
      \ifkv@handled@
      \else
        \kv@error@unknownkey{#1}{#2}%
      \fi
    \fi
  \else
    \ifx\kv@value\relax
&     \unless\ifcsname KV@#1@#2@default\endcsname
$     \begingroup\expandafter\expandafter\expandafter\endgroup
$     \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
        \kv@error@novalue{#1}{#2}%
      \else
        \csname KV@#1@#2@default\endcsname
        \relax
      \fi
    \else
      \csname KV@#1@#2\endcsname{#3}%
    \fi
  \fi
}
```

\kv@processor@known

```
\def\kv@processor@known#1#2#3{%
  \begingroup
    \csname @safe@activestrue\endcsname
    \let\ifincsname\iftrue
    \edef\KVS@temp{\endgroup
      \noexpand\KVS@ProcessorKnown{#1}\noexpand#2{#3}%
    }%
  \KVS@temp
}
```

\KVS@ProcessorKnown

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

```
   \KVS@AddUnhandled#2{#3}{#4}%
  \else
   \kv@handled@true
   \csname KVS@#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{%
&  \ifx#1\KVS@empty
&  \else
&   \unexpanded\expandafter{#1},%
&  \fi
&  \unexpanded{{#2}={#3}}%
& }%
$ \begingroup
$  \ifx#1\KVS@empty
$   \toks@{{#2}={#3}}%
$  \else
$   \toks@\expandafter{#1,{#2}={#3}}%
$  \fi
$  \xdef\KVS@Global{\the\toks@}%
$ \endgroup
$ \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`

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

`\kv@error@unknownkey`

```
\def\kv@error@unknownkey{%
  \kv@error@generic{Undefined}%
}
```

`\kv@error@generic`

```
\def\kv@error@generic#1#2#3{%
  \PackageError{kvsetkeys}{%
    #1 key `#3'%
  }{%
    The keyval family of the key `#3' is `#2'.\MessageBreak
    The setting of the key is ignored because of the error.\MessageBreak
    \MessageBreak
    \@ehc
  }%
}
```

3.10 Do it all

`\kvsetkeys`

```
\long\def\kvsetkeys#1#2{%
  \kv@parse{#2}{\kv@processor@default{#1}}%
}
```

`\kvsetkeys@expandafter`

```
\def\kvsetkeys@expandafter#1#2{%
  \expandafter\kv@parse\expandafter{#2}{%
    \kv@processor@default{#1}%
  }%
}
```

`\KVS@cmd`

```
\def\KVS@cmd{0}%
```

`\KVS@cmd@inc`

```
\def\KVS@cmd@inc{%
& \edef\KVS@cmd{\the\numexpr\KVS@cmd+1}%
$ \begingroup
$   \count255=\KVS@cmd\relax
$   \advance\count255 by 1\relax
$ \edef\x{\endgroup
$   \noexpand\def\noexpand\KVS@cmd{\number\count255}%
$ }%
$ \x
}
```

`\KVS@cmd@dec`

```
\def\KVS@cmd@dec{%
& \edef\KVS@cmd{\the\numexpr\KVS@cmd-1}%
$ \begingroup
$   \count255=\KVS@cmd\relax
$   \advance\count255 by -1\relax
$ \edef\x{\endgroup
$   \noexpand\def\noexpand\KVS@cmd{\number\count255}%
$ }%
$ \x
}
```

`\KVS@empty`

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

`\kvsetknownkeys`

```
\def\kvsetknownkeys{%
  \expandafter
  \KVS@setknownkeys\csname KVS@cmd\KVS@cmd\endcsname{}%
}
```

`\KVS@setknownkeys`

```
\long\def\KVS@setknownkeys#1#2#3#4#5{%
  \let#1\KVS@empty
  \KVS@cmd@inc
  #2\kv@parse#2{#5}{\kv@processor@known{#3}#1}%
  \KVS@cmd@dec
  \let#4=#1%
}
```

`\kvsetknownkeys@expandafter`

```
\def\kvsetknownkeys@expandafter{%
  \expandafter
  \KVS@setknownkeys
      \csname KVS@cmd\KVS@cmd\endcsname\expandafter
}
```

```
\KVS@AtEnd%
⟨/package⟩
```

4 Installation

4.1 Download

Package. This package is available on CTAN[1]:

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

`CTAN:macros/latex/contrib/kvsetkeys/kvsetkeys.pdf` Documentation.

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:

```
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.

---

[1] `CTAN:pkg/kvsetkeys`

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:

```
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 pdfLaTeX:

```
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-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 ε-TeX.

[2016/05/16 v1.17]

- Documentation updates.

[2019/12/15 v1.18]

- Documentation updates.
- Avoid etexcmds and infwarwerr 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

A

`\advance` 540, 550
`\aftergroup` 76

B

`\begin` 34, 35

C

`\catcode` 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
`\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, 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

`\define@key` 29
`\documentclass` 2
`\dots` 38, 41, 43

E

\empty 64, 65
\end 45, 46
\endcsname 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
\endinput 76, 161
\endlinechar 51, 82, 118, 124, 136

I

\ifcsname 406, 409, 424, 448, 451, 466
\iffalse 389
\ifincsname 398, 440, 500, 507
\ifkv@handled@ 389, 392, 394, 417, 459
\iftrue 392, 398, 440, 500, 507
\ifx 62, 65, 68, 97, 105, 108, 163, 166, 174, 177, 193, 283, 408, 411, 423, 426, 450, 453, 465, 468, 481, 488
\immediate 70, 99
\input 167

K

\kill 36
\kv@break *4*, 363
\kv@error@generic 512, 515, 517
\kv@error@novalue 427, 469, 511
\kv@error@unknownkey 412, 419, 514
\kv@fam 405, 447
\kv@handled@false 388
\kv@handled@true 391, 394, 414, 456
\kv@key 342
\kv@list 211, 324
\kv@normalize *3*, 199, 323
\kv@parse *3*, 322, 528, 531, 564
\kv@parse@normalized *3*, 324, 326
\kv@processor@default *4*, 395, 528, 532
\kv@processor@known *4*, 437, 564
\kv@set@family@handler *5*, 20, 497
\kv@unset@family@handler *5*, 504
\kv@value 344, 351, 356, 423, 465
\KVS@@Comma 230, 232, 236
\KVS@@CommaComma 264, 266
\KVS@@CommaSpace 254, 256
\KVS@@Equals 279, 281, 295
\KVS@@EqualsSpace 313, 315
\KVS@@Process 347, 350
\KVS@@SpaceComma 242, 246
\KVS@@SpaceEquals 301, 305
\KVS@AddUnhandled 454, 461, 479
\KVS@AtEnd 142, 143, 161, 573
\KVS@break 360, 364, 386
\KVS@cmd 535, 537, 539, 542, 547, 549, 552, 559, 571
\KVS@cmd@dec 546, 565
\KVS@cmd@inc 536, 563
\KVS@Comma 202, 216, 224
\KVS@CommaComma 205, 219, 263
\KVS@CommaParse 371, 373
\KVS@CommaSpace 204, 218, 253
\KVS@Empty 184, 193, 283
\KVS@empty 481, 488, 556, 562
\KVS@Equals 206, 273
\KVS@EqualsSpace 208, 312
\KVS@FirstOfTwo 185, 194, 284
\KVS@Global 209, 211, 220, 222, 354, 356, 493, 495
\KVS@IfEmpty 187, 234, 247, 257, 267, 293, 306, 316, 330, 335, 343, 374, 380
\KVS@MaybeBreak 334, 341, 359, 361, 364, 379, 386
\KVS@Nil 230, 232, 236, 242, 246, 250, 254, 256, 260, 264, 266, 270, 279, 281, 295, 301, 305, 309, 313, 315, 319, 327, 329, 332, 337, 340, 347, 350, 371, 373, 382
\KVS@Parse 327, 329
\KVS@Process 332, 340
\KVS@ProcessorDefault 400, 404
\KVS@ProcessorKnown 442, 446
\KVS@SecondOfTwo 186, 196, 286
\KVS@setknownkeys 559, 561, 570
\KVS@SpaceComma 203, 217, 240
\KVS@SpaceEquals 207, 299
\KVS@Temp 188, 191, 193, 282, 283
\KVS@temp 399, 402, 441, 444
\kvsetkeys *5*, 14, 527
\kvsetkeys@expandafter *6*, 530
\kvsetknownkeys *5*, 557
\kvsetknownkeys@expandafter 568

L

\lccode 225, 226, 274, 275
\lowercase 227, 276

M

\makeatletter 7
\makeatother 32
\mbox 36
\MessageBreak 521, 522, 523

N

\newcommand 8
\number 542, 552
\numexpr 177, 537, 547

P

\PackageError 171, 518
\PackageInfo 73
\ProvidesPackage 66, 114

Q

\qquad 36

S

\space 25

T

\tag 8, 37, 39, 40, 42, 44
\textgreater 16
\textless 16
\texttt 15

\the 16, 24, 124, 125, 126, 127, 128, 129, 130, 131, 144, 191, 209, 220, 230, 233, 242, 254, 264, 279, 282, 291, 301, 313, 354, 493, 537, 547

\TMP@EnsureCode 141, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160

\TMP@RequirePackage 164, 170

\toks@ 12, 16, 23, 24, 175, 190, 191, 201, 209, 215, 220, 229, 230, 233, 242, 248, 254, 258, 264, 268, 278, 279, 282, 289, 291, 301, 307, 313, 317, 353, 354, 489, 491, 493

\toksdef 175

U

\unexpanded 188, 351, 483, 485

\unless . . . 406, 409, 424, 448, 451, 466

\usepackage 3, 4, 5

W

\write 70, 99

X

\x . . . 61, 62, 65, 69, 73, 75, 98, 103, 113, 122, 134, 541, 544, 551, 554