

The kvsetkeys package

Heiko Oberdiek*

2016/05/16 v1.17

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

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

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

1 Documentation

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

1.1 Motivation

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

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

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

```

\setkeys{fam}{key={{value}}}  

\setkeys{fam}{key={{value}}}  

\setkeys{fam}{key= {{value}}}  


```

This package `kvsetkeys` follows a much stronger rule: Exactly one level of braces are removed from an item, if the item is surrounded by curly braces. An item can be a the key value pair, the key or the value.

```

\kvsetkeys{fam}{key={value}} → 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 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 `⟨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 2016/05/16 v1.17 `\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 2016/05/16 v1.17 `\kv@processor@known` also defines macro `\kv@fam` with meaning `<family>` for convenience.

1.5 Default family handler

`\kv@processor@default` calls `<handler>`, the default handler for the family, if the key does not exist in the family. The handler is called with two arguments, the key and the value. It can be defined with `\kv@set@family@handler`:

`\kv@set@family@handler {<family>} {<handler definition>}`

This sets the default family handler for the keyval family `<family>`. Inside `<handler definition>` `#1` stands for the key and `#2` is the value. Also `\kv@key` and `\kv@value` can be used for the key and the value. If the value is not given, `\kv@value` has the meaning `\relax`.

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

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

1.6 Put it all together

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

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

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

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

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

Pseudo code:

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

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

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

Thus you can replace `\setkeys` of package `keyval` by the key value parser of this package:

```
\renewcommand*\setkeys{\kvsetkeys}
or
\let\setkeys\kvsetkeys
```

1.7 Comma separated lists

Since version 2007/09/29 v1.3 this package also supports the normalizing and parsing of general comma separated lists.

`\comma@normalize {⟨comma list⟩}`

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

`\comma@parse {⟨comma list⟩} {⟨processor⟩}`

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

`\comma@parse@normalized {⟨normalized comma list⟩} {⟨processor⟩}`

The list is parsed. Empty entries are ignored. `⟨processor⟩` is called for each non-empty entry with the entry as argument:

`⟨processor⟩{⟨entry⟩}`

Also the entry is stored in the macro `\comma@entry`.

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

2 Example

The following example prints a short piece of HTML code using the tabbing environment for indenting purpose and a key value syntax for specifying the attributes of an HTML tag. The example illustrates the use of a default family handler.

```
1  $\langle *example \rangle$ 
2 \documentclass{article}
3 \usepackage[T1]{fontenc}
4 \usepackage{kvsetkeys}
5 \usepackage{keyval}
6
7 \makeatletter
8 \newcommand*{\tag}[2][ ]{%
9   % #1: attributes
10  % #2: tag name
11  \begingroup
12    \toks@={}%
13    \let\@endslash\@empty
14    \kvsetkeys{tag}{#1}%
15    \texttt{%
16      \textless #2\the\toks@\@endslash\textgreater
17    }%
18  \endgroup
19 }
20 \kv@set@family@handler{tag}{%
21   % #1: key
22   % #2: value
23   \toks@\expandafter{%
24     \the\toks@
25     \space
26     #1=\string"#2\string"%
27   }%
28 }
29 \define@key{tag}{/}[ ]{%
30   \def\@endslash{/}%
31 }
32 \makeatother
33
34 \begin{document}
35 \begin{tabbing}
36   \mbox{} \quad \quad \quad \kill
37   \tag{html} \backslash
38   \backslash \dots \backslash
39   \backslash \tag[border=1]{table} \backslash \backslash
40   \backslash \backslash \tag[width=200, span=3, /]{colgroup} \backslash \backslash
41   \backslash \backslash \dots \backslash \backslash
42   \backslash \tag{/table} \backslash \backslash
43   \backslash \dots \backslash \backslash
44   \tag{/html} \backslash \backslash
45 \end{tabbing}
46 \end{document}
```

47 `\example`

3 Implementation

3.1 Identification

48 `(*package)`

Reload check, especially if the package is not used with L^AT_EX.

```
49 \begingroup\catcode61\catcode48\catcode32=10\relax%
50 \catcode13=5 % ^~M
51 \endlinechar=13 %
52 \catcode35=6 % #
53 \catcode39=12 % '
54 \catcode44=12 % ,
55 \catcode45=12 % -
56 \catcode46=12 % .
57 \catcode58=12 % :
58 \catcode64=11 % @
59 \catcode123=1 % {
60 \catcode125=2 % }
61 \expandafter\let\expandafter\x\csname ver@kvsetkeys.sty\endcsname
62 \ifx\x\relax % plain-TeX, first loading
63 \else
64 \def\empty{}%
65 \ifx\x\empty % LaTeX, first loading,
66 % variable is initialized, but \ProvidesPackage not yet seen
67 \else
68 \expandafter\ifx\csname PackageInfo\endcsname\relax
69 \def\x#1#2{%
70 \immediate\write-1{Package #1 Info: #2.}%
71 }%
72 \else
73 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
74 \fi
75 \x{kvsetkeys}{The package is already loaded}%
76 \aftergroup\endinput
77 \fi
78 \fi
79 \endgroup%
```

Package identification:

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



```

97 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
98 \def\x#1#2#3[#4]{\endgroup
99 \immediate\write-1{Package: #3 #4}%
100 \xdef#1{#4}%
101 }%
102 \else
103 \def\x#1#2[#3]{\endgroup
104 #2[{#3}]%
105 \ifx#1\@undefined
106 \xdef#1{#3}%
107 \fi
108 \ifx#1\relax
109 \xdef#1{#3}%
110 \fi
111 }%
112 \fi
113 \expandafter\x\csname ver@kvsetkeys.sty\endcsname
114 \ProvidesPackage{kvsetkeys}%
115 [2016/05/16 v1.17 Key value parser (HO)]%
116 \begingroup\catcode61\catcode48\catcode32=10\relax%
117 \catcode13=5 % ^M
118 \endlinechar=13 %
119 \catcode123=1 % {
120 \catcode125=2 % }
121 \catcode64=11 % @
122 \def\x{\endgroup
123 \expandafter\edef\csname KVS@AtEnd\endcsname{%
124 \endlinechar=\the\endlinechar\relax
125 \catcode13=\the\catcode13\relax
126 \catcode32=\the\catcode32\relax
127 \catcode35=\the\catcode35\relax
128 \catcode61=\the\catcode61\relax
129 \catcode64=\the\catcode64\relax
130 \catcode123=\the\catcode123\relax
131 \catcode125=\the\catcode125\relax
132 }%
133 }%
134 \x\catcode61\catcode48\catcode32=10\relax%
135 \catcode13=5 % ^M
136 \endlinechar=13 %
137 \catcode35=6 % #
138 \catcode64=11 % @
139 \catcode123=1 % {
140 \catcode125=2 % }
141 \def\TMP@EnsureCode#1#2{%
142 \edef\KVS@AtEnd{%
143 \KVS@AtEnd
144 \catcode#1=\the\catcode#1\relax
145 }%
146 \catcode#1=#2\relax
147 }
148 \TMP@EnsureCode{36}{3}% $
149 \TMP@EnsureCode{38}{4}% &
150 \TMP@EnsureCode{39}{12}% '
151 \TMP@EnsureCode{43}{12}% +
152 \TMP@EnsureCode{44}{12}% ,
153 \TMP@EnsureCode{45}{12}% -
154 \TMP@EnsureCode{46}{12}% .

```

```

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

```

3.2 Package loading

```

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

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

¹CTAN:pkg/kvsetkeys

[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_EX Files” ([CTAN:pkg/tds](#)). Directories with `texmf` in their name are usually organized this way.

4.2 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
```

4.3 Package installation

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

```
tex kvsetkeys.dtx
```

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

<code>kvsetkeys.sty</code>	→ <code>tex/generic/oberdiek/kvsetkeys.sty</code>
<code>kvsetkeys.pdf</code>	→ <code>doc/latex/oberdiek/kvsetkeys.pdf</code>
<code>kvsetkeys-example.tex</code>	→ <code>doc/latex/oberdiek/kvsetkeys-example.tex</code>
<code>kvsetkeys.dtx</code>	→ <code>source/latex/oberdiek/kvsetkeys.dtx</code>

If you have a `docstrip.cfg` that configures and enables `docstrip`’s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

4.4 Refresh file name databases

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

4.5 Some details for the interested

Unpacking with L^AT_EX. The `.dtx` chooses its action depending on the format:

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

L^AT_EX: Generate the documentation.

If you insist on using L^AT_EX for `docstrip` (really, `docstrip` does not need L^AT_EX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{kvsetkeys.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL^AT_EX:

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

5 References

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

6 History

[2006/03/06 v1.0]

- First version.

[2006/10/19 v1.1]

- Fix of `\kv@set@family@handler`.
- Example added.

[2007/09/09 v1.2]

- Using package `infwarerr` for error messages.
- Catcode section rewritten.

[2007/09/29 v1.3]

- Normalizing and parsing of comma separated lists added.
- `\kv@normalize` rewritten.
- Robustness increased for normalizing and parsing, e.g. for values with unmatched conditionals.
- ε -T_EX is used if available.
- Tests added for normalizing and parsing.

[2009/07/19 v1.4]

- Bug fix for `\kv@normalize`: unwanted space removed (Florent Chervet).

[2009/07/30 v1.5]

- Documentation addition: recommendation for Joseph Wright’s review article.

[2009/12/12 v1.6]

- Short info shortened.

[2009/12/22 v1.7]

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

[2010/01/28 v1.8]

- Compatibility to `iniTeX` added.

[2010/03/01 v1.9]

- Support of `\par` inside values.

[2011/01/30 v1.10]

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

[2011/03/03 v1.11]

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

[2011/04/05 v1.12]

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

[2011/04/07 v1.13]

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

[2011/06/15 v1.14]

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

[2011/10/18 v1.15]

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

[2012/04/25 v1.16]

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

[2016/05/16 v1.17]

- Documentation updates.

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		450, 461, 464, 468, 479, 482, 486, 510, 513, 517, 520, 570, 582
<code>\\$</code>	189, 192	
<code>\&</code>	190, 193	
<code>\,</code>	236, 237	
<code>\=</code>	36, 285, 286	
<code>\></code>	38, 39, 40, 41, 42, 43	
<code>\@PackageError</code>	529	
<code>\@UnDeFiNeD</code>	520	
<code>\@ehc</code>	535	
<code>\@empty</code>	13	
<code>\@endslash</code>	13, 16, 30	
<code>\@undefined</code>	105	
<code>\%</code>	37, 38, 39, 40, 41, 42, 43, 44	
<code>\~</code>	237, 286	
A		
<code>\advance</code>	551, 561	
<code>\aftergroup</code>	76	
B		
<code>\begin</code>	34, 35	
C		
<code>\catcode</code> .	49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 80, 81, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 116, 117, 119, 120, 121, 125, 126, 127, 128, 129, 130, 131, 134, 135, 137, 138, 139, 140, 144, 146, 189, 190, 192, 193	
<code>\comma@break</code>	7, <u>396</u>	
<code>\comma@entry</code>	387	
<code>\comma@list</code>	233, 379	
<code>\comma@normalize</code>	6, <u>224</u> , 378	
<code>\comma@parse</code>	6, <u>377</u>	
<code>\comma@parse@normalized</code> .	6, 379, <u>381</u>	
<code>\count</code>	550, 551, 553, 560, 561, 563	
<code>\csname</code>	61, 68, 97, 113, 123, 163, 166, 176, 181, 183, 408, 419, 422, 426, 437, 440, 444,	
D		
<code>\define@key</code>	29	
<code>\documentclass</code>	2	
<code>\dots</code>	38, 41, 43	
E		
<code>\empty</code>	64, 65	
<code>\end</code>	45, 46	
<code>\endcsname</code> . . .	61, 68, 97, 113, 123, 163, 166, 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	
<code>\endinput</code>	76, 161	
<code>\endlinechar</code>	51, 82, 118, 124, 136	
<code>\etex@unexpanded</code> . .	199, 362, 494, 496	
I		
<code>\ifcase</code>	180	
<code>\ifcsname</code> .	417, 420, 435, 459, 462, 477	
<code>\ifetex@unexpanded</code>	180	
<code>\iffalse</code>	400	
<code>\ifincsname</code>	409, 451, 511, 518	
<code>\ifkv@handled@</code>	400, 403, <u>405</u> , 428, 470	
<code>\iftrue</code>	403, 409, 451, 511, 518	
<code>\ifx</code>	62, 65, 68, 97, 105, 108, 163, 166, 176, 181, 183, 204, 294, 419, 422, 434, 437, 461, 464, 476, 479, 492, 499	
<code>\immediate</code>	70, 99	
<code>\input</code>	167	
K		
<code>\kill</code>	36	
<code>\kv@break</code>	4, <u>374</u>	
<code>\kv@error@generic</code>	523, 526, <u>528</u>	
<code>\kv@error@novalue</code>	438, 480, <u>522</u>	

\KV@error@unknownkey	..	423 , 430 , 525	\KVS@Temp	199 , 202 , 204 , 293 , 294
\kv@fam		416 , 458	\KVS@temp	410 , 413 , 452 , 455
\kv@handled@false	399	\kvsetkeys	5 , 14 , 538
\kv@handled@true	..	402 , 405 , 425 , 467	\kvsetkeys@expandafter	6 , 541
\kv@key	353	\kvsetknownkeys	6 , 568
\kv@list	222 , 335	\kvsetknownkeys@expandafter	579
\kv@normalize	3 , 210 , 334			
\kv@parse	3 , 333 , 539 , 542 , 575	L		
\kv@parse@normalized	4 , 335 , 337	\lccode	236 , 237 , 285 , 286
\kv@processor@default	4 , 406 , 539 , 543		\lowercase	238 , 287
\kv@processor@known	5 , 448 , 575	M		
\kv@set@family@handler	... 5 , 20 , 508		\makeatletter	7
\kv@unset@family@handler	5 , 515	\makeatother	32
\kv@value	355 , 362 , 367 , 434 , 476	\mbox	36
\KVS@@Comma	241 , 243 , 247	\MessageBreak	532 , 533 , 534
\KVS@@CommaComma	275 , 277	N		
\KVS@@CommaSpace	265 , 267	\newcommand	8
\KVS@@Equals	290 , 292 , 306	\number	553 , 563
\KVS@@EqualsSpace	324 , 326	\numexpr	548 , 558
\KVS@@Process	358 , 361	P		
\KVS@@SpaceComma	253 , 257	\PackageInfo	73
\KVS@@SpaceEquals	312 , 316	\ProvidesPackage	66 , 114
\KVS@AddUnhandled	465 , 472 , 490	Q		
\KVS@AtEnd	142 , 143 , 161 , 584	\qqquad	36
\KVS@break	371 , 375 , 397	R		
\KVS@cmd	546 , 548 ,	\RequirePackage	173 , 174
	550 , 553 , 558 , 560 , 563 , 570 , 582		S		
\KVS@cmd@dec	557 , 576	\space	25
\KVS@cmd@inc	547 , 574	T		
\KVS@Comma	213 , 227 , 235	\tag	8 , 37 , 39 , 40 , 42 , 44
\KVS@CommaComma	216 , 230 , 274	\textgreater	16
\KVS@CommaParse	382 , 384	\textless	16
\KVS@CommaSpace	215 , 229 , 264	\texttt	15
\KVS@Empty	195 , 204 , 294	\the	16 , 24 , 124 ,
\KVS@empty	492 , 499 , 567 , 573		125 , 126 , 127 , 128 , 129 , 130 ,	
\KVS@Equals	217 , 284		131 , 144 , 202 , 220 , 231 , 241 ,	
\KVS@EqualsSpace	219 , 323		244 , 253 , 265 , 275 , 290 , 293 ,	
\KVS@FirstOfTwo	196 , 205 , 295		302 , 312 , 324 , 365 , 504 , 548 , 558	
\KVS@Global	220 ,	\TMP@EnsureCode	141 ,
	222 , 231 , 233 , 365 , 367 , 504 , 506			148 , 149 , 150 , 151 , 152 , 153 ,	
\KVS@ifEmpty			154 , 155 , 156 , 157 , 158 , 159 , 160	
	198 , 245 , 258 , 268 , 278 , 304 ,		\TMP@RequirePackage	...	164 , 170 , 171
	317 , 327 , 341 , 346 , 354 , 385 , 391		\toks@	12 , 16 ,
\KVS@MaybeBreak			23 , 24 , 177 , 201 , 202 , 212 , 220 ,	
	345 , 352 , 370 , 372 , 375 , 390 , 397			226 , 231 , 240 , 241 , 244 , 253 ,	
\KVS@Nil	241 , 243 , 247 , 253 , 257 , <				

	W	X
\write	70, 99	61, 62, 65, 69, 73, 75, 98, 103, 113, 122, 134, 552, 555, 562, 565