

The kvoptions package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2009/07/17 v3.2

Abstract

This package is intended for package authors who want to use options in key value format for their package options.

Contents

1	Introduction	2
1.1	The beginning	2
1.2	Overview	3
2	Usage	3
2.1	Process options	3
2.1.1	\ProcessKeyvalOptions	3
2.1.2	\ProcessLocalKeyvalOptions	3
2.1.3	\SetupKeyvalOptions	4
2.2	Option declarations	4
2.2.1	\DeclareStringOption	4
2.2.2	\DeclareBoolOption	5
2.2.3	\DeclareComplementaryOption	5
2.2.4	\DeclareVoidOption	6
2.2.5	\DeclareDefaultOption	6
2.2.6	Dynamic options	6
2.2.7	\DisableKeyvalOption	7
2.3	Summary of internal macros	7
2.4	plain-T _E X	8
3	Example	8
4	Package options	10
4.1	Package kvoptions-patch	10
4.2	Option debugshow	11
5	Limitations	11
5.1	Compatibility	11
5.1.1	Package kvoptions-patch vs. package xkvltxp	11
5.2	Limitations	12
5.2.1	Option comparisons	12
5.2.2	Option list parsing with package kvoptions-patch	12
6	Implementation	12
6.1	Preamble	12
6.2	Option declaration macros	15
6.2.1	\SetupKeyvalOptions	15
6.2.2	\DeclareBoolOption	16
6.2.3	\DeclareStringOption	18

6.2.4	<code>\DeclareVoidOption</code>	19
6.2.5	<code>\DeclareDefaultOption</code>	20
6.3	Dynamic options	20
6.3.1	<code>\DisableKeyvalOption</code>	20
6.4	Process options	22
6.5	<code>\ProcessKeyvalOptions</code>	22
6.6	<code>\ProcessLocalKeyvalOptions</code>	24
6.6.1	Helper macros	25
6.7	plain- \TeX	26
6.8	Package <code>kvoptions-patch</code>	26
7	Test	34
7.1	Preface for standard catcode check	34
7.2	Catcode checks for loading	34
8	Installation	36
8.1	Download	36
8.2	Bundle installation	36
8.3	Package installation	36
8.4	Refresh file name databases	37
8.5	Some details for the interested	37
9	References	37
10	History	38
	[0000/00/00 v0.0]	38
	[2004/02/22 v1.0]	38
	[2006/02/16 v2.0]	38
	[2006/02/20 v2.1]	38
	[2006/06/01 v2.2]	38
	[2006/08/17 v2.3]	39
	[2006/08/22 v2.4]	39
	[2007/04/11 v2.5]	39
	[2007/05/06 v2.6]	39
	[2007/06/11 v2.7]	39
	[2007/10/02 v2.8]	39
	[2007/10/11 v2.9]	39
	[2007/10/18 v3.0]	39
	[2009/04/10 v3.1]	39
	[2009/07/17 v3.2]	39
11	Index	39

1 Introduction

1.1 The beginning

This package addresses class or package writers that want to allow their users to specify options as key value pairs, e.g.:

```
\documentclass[verbose=false,name=me]{myclass}
\usepackage[format=print]{mylayout}
```

Prominent example is package `hyperref`, probably the first package that offers this service. It's `\ProcessOptionsWithKV` is often copied und used in other packages, e.g. package `helvet` that uses this interface for its option `scaled`.

However copying code is not the most modern software development technique. And `hyperref`'s code for `\ProcessOptionsWithKV` was changed to fix bugs. The version used in other packages depends on the time of copying and the awareness

of `hyperref`'s changes. Now the code is sourced out into this package and available for other package or class writers.

1.2 Overview

Package `kvoptions` connects package `keyval` with \LaTeX 's package and class *options*:

Package <code>keyval</code>	Package <code>kvoptions</code>	\LaTeX kernel
<code>\define@key</code>	<code>\DeclareVoidOption</code> <code>\DeclareStringOption</code> <code>\DeclareBoolOption</code> <code>\DeclareComplementaryOption</code> <code>\DisableKeyvalOption</code>	<code>\DeclareOption</code>
	<code>\DeclareDefaultOption</code>	<code>\DeclareOption*</code>
	<code>\ProcessKeyvalOptions</code>	<code>\ProcessOptions*</code>
	Option patch	Class/package option system
	<code>\SetupKeyvalOptions</code>	

2 Usage

2.1 Process options

2.1.1 `\ProcessKeyvalOptions`

<code>\ProcessKeyvalOptions {<i><family></i>}</code> <code>\ProcessKeyvalOptions *</code>
--

This command evaluates the global or local options of the package that are defined with `keyval`'s interface within the family *<family>*. It acts the same way as \LaTeX 's `\ProcessOptions*`. In a package unknown global options are ignored, in a class they are added to the unknown option list. The known global options and all local options are passed to `keyval`'s `\setkeys` command for executing the options. Unknown options are reported to the user by an error.

If the family name happens to be the same as the name of the package or class where `\ProcessKeyvalOptions` is used or the family name has previously been setup by `\SetupKeyvalOptions`, then `\ProcessKeyvalOptions` knows the family name already and you can use the star form without mandatory argument.

2.1.2 `\ProcessLocalKeyvalOptions`

<code>\ProcessLocalKeyvalOptions {<i><family></i>}</code> <code>\ProcessLocalKeyvalOptions *</code>
--

This macro has the same syntax and works similar as `\ProcessKeyvalOptions`. However it ignores global options and only processes the local package options. Therefore it only can be used inside a package. An error is thrown, if it is used inside a class.

Neither of the following macros are necessary for `\ProcessKeyvalOptions`. They just help the package/class author in common tasks.

2.1.3 \SetupKeyvalOptions

```
\SetupKeyvalOptions {
  family = <family>,
  prefix = <prefix>
}
```

This command allows to configure the default assumptions that are based on the current package or class name. L^AT_EX remembers this name in `\@currname`. The syntax description of the default looks a little weird, therefor an example is given for a package or class named `foobar`.

Key	Default	(example)	Used by
family	<code>\@currname</code>	(foobar)	<code>\ProcessKeyvalOptions*</code> <code>\DeclareBoolOption</code> <code>\DeclareStringOption</code>
prefix	<code>\@currname</code> @	(foobar@)	<code>\DeclareBoolOption</code> <code>\DeclareStringOption</code> <code>\DeclareVoidOption</code>

2.2 Option declarations

The options for `\ProcessKeyvalOptions` are defined by `keyval`'s `\define@key`. Common purposes of such keys are boolean switches, they enable or disable something. Or they store a name or some kind of string in a macro. The following commands help the user. He declares what he wants and `kvoptions` take care of the key definition, resource allocation and initialization.

In order to avoid name clashes of macro names, internal commands are prefixed. Both the prefix and the family name for the defined keys can be configured by `\SetupKeyvalOptions`.

2.2.1 \DeclareStringOption

```
\DeclareStringOption [<init>] {<key>} [<default>]
```

A macro is created that remembers the value of the key `<key>`. The name of the macro consists of the option name `<key>` that is prefixed by the prefix (see 2.1.3). The initial contents of the macro can be given by the first optional argument `<init>`. The default is empty.

The the option `<key>` is defined. The option code just stores its value in the macro. If the optional argument at the end of `\DeclareStringOption` is given, then option `<key>` is defined with the default `<default>`.

Example for a package with the following two lines:

```
\ProvidesPackage{foobar}
\DeclareStringOption[me]{name}
```

Then `\DeclareStringOption` defines the macro with content `me`, note L^AT_EX complains if the name of the macro already exists:

```
\newcommand*{\foobar@name}{me}
```

The option definition is similar to:

```
\define@key{foobar}{name}{%
  \renewcommand*{\foobar@name}{#1}%
}
```

2.2.2 \DeclareBoolOption

`\DeclareBoolOption [<init>] {<key>}`

A boolean switch is generated, initialized by value *<init>* and the corresponding key *<key>* is defined. If the initialization value is not given, `false` is used as default.

The internal actions of `\DeclareBoolOption` are shown below. The example is given for a package author who has the following two lines in his package/class:

```
\ProvidesPackage{foobar}  
\DeclareBoolOption{verbose}
```

First a new switch is created:

```
\newif\iffoobar@verbose
```

and initialized:

```
\foobar@verbosefalse
```

Finally the key is defined:

```
\define@key{foobar}{verbose}[true]{...}
```

The option code configures the boolean option in the following way: If the author specifies `true` or `false` then the switch is turned on or off respectively. Also the option can be given without explicit value. Then the switch is enabled. Other values are reported as errors.

Now the switch is ready to use in the package/class, e.g.:

```
\iffoobar@verbose  
% print verbose message  
\else  
% be quiet  
\fi
```

Users of package `\ifthen` can use the switch as boolean:

```
\boolean{foobar@verbose}
```

2.2.3 \DeclareComplementaryOption

`\DeclareComplementaryOption {<key>} {<parent>}`

Sometimes contrasting names are used to characterize the two states of a boolean switch, for example `draft` vs. `final`. Both options behave like boolean options but they do not need to different switches, they should share one. `\DeclareComplementaryOption` allows this. The option *<key>* shares the switch of option *<parent>*. Example:

```
\DeclareBoolOption{draft}  
\DeclareComplementaryOption{final}{draft}
```

Then `final` sets the switch of `draft` to `false`, and `final=false` enables the `draft` switch.

2.2.4 `\DeclareVoidOption`

`\DeclareVoidOption{<key>}{<code>}`

`\ProcessKeyvalOptions` can be extended to recognize options that are declared in traditional way by `\DeclareOption`. But in case of the error that the user specifies a value, then this option would not be recognized as key value option because of `\DeclareOption` and not detected as traditional option because of the value part. The user would get an unknown option error, difficult to understand.

`\DeclareVoidOption` solves this problem. It defines the option `<key>` as key value option. If the user specifies a value, a warning is given and the value is ignored.

The code part `<code>` is stored in a macro. The name of the macro consists of the option name `<key>` that is prefixed by the prefix (see 2.1.3). If the option is set, the macro will be executed. During the execution `\CurrentOption` is available with the current key name.

2.2.5 `\DeclareDefaultOption`

`\DeclareDefaultOption{<code>}`

This command does not define a specific key, it is the equivalent to L^AT_EX's `\DeclareOption*`. It allows the specification of a default action `<code>` that is invoked if an unknown option is found. While `<code>` is called, macro `\CurrentOption` contains the current option string. In addition `\CurrentOptionValue` contains the value part if the option string is parsable as key value pair, otherwise it is `\relax`. `\CurrentOptionKey` contains the key of the key value pair, or the whole option string, if it misses the equal sign.

Inside packages typical default actions are to pass unknown options to another package. Or an error message can be thrown by `\@unknownoptionerror`. This is the original error message that L^AT_EX gives for unknown package options. This error message is easier to understand for the user as the error message from package `keyval` that is given otherwise.

A Class ignores unknown options and puts them on the unused option list. Let L^AT_EX do the job and just call `\OptionNotUsed`. Or the options can be passed to another class that is later loaded.

2.2.6 Dynamic options

Options of L^AT_EX's package/class system are cleared in `\ProcessOptions`. They modify the static model of a package. For example, depending on option `bookmarks` package `hyperref` loads differently.

Options, however, defined by `keyval`'s `\define@key` remain defined, if the options are processed by `\setkeys`. Therefore these options can also be used to model the dynamic behaviour of a package. For example, in `hyperref` the link colors can be changed everywhere until the end in `\end{document}`.

However package `color` that adds color support is necessary and it cannot be loaded after `\begin{document}`. Option `colorlinks` that loads `color` should be active until `\begin{document}` and die in some way if it is too late for loading packages. With `\DisableKeyvalOption` the package/class author can specify and configure the death of an option and controls the life period of the option.

2.2.7 \DisableKeyvalOption

```
\DisableKeyvalOption [<options>] {<family>} {<key>}  
<options>:  
  action          = undef, warning, error, or ignore    default: undef  
  global or local                                default: global  
  package or class = <name>
```

`\DisableKeyvalOption` can be called to mark the end when the option *<key>* is no longer useful. The behaviour of an option after its death can be configured by action:

undef: The option will be undefined, If it is called, `\setkeys` reports an error because of unknown key.

error or warning: Use of the option will cause an error or warning message. Also these actions require that exclusively either the package or class name is given in options package or class.

ignore: The use of the option will silently be ignored.

The option's death can be limited to the end of the current group, if option `local` is given. Default is `global`.

The package/class author can wish the end of the option already during the package loading, then he will have static behaviour. In case of dynamic options `\DisableKeyvalOptions` can be executed everywhere, also outside the package. Therefore the family name and the package/class name is usually unknown for `\DisableKeyvalOptions`. Therefore the argument for the family name is mandatory and for some actions the package/class name must be provided.

Usually a macro would configure the option death, Example:

```
\ProvidesPackage{foobar}  
\DeclareBoolOption{color}  
\DeclareStringOption[red]{emphcolor}  
\ProcessKeyvalOptions*  
  
\newcommand*{\foobar@DisableOption}[2]{%  
  \DisableKeyvalueOption[  
    action={#1},  
    package=foobar  
  ]{foobar}{#2}%  
}  
  
\iffobar@color  
  \RequirePackage{color}  
  \renewcommand*{\emph}[1]{\textcolor{\foobar@emphcolor}{#1}}  
\else  
  % Option emphcolor is not wrong, if we have color support.  
  % otherwise the option has no effect, but we don't want to  
  % remove it. Therefore action 'ignore' is the best choice:  
  \foobar@DisableOption{ignore}{emphcolor}  
\fi  
% No we don't need the option 'color'.  
\foobar@DisableOption{warning}{color}  
  
% With color support option 'emphcolor' will dynamically  
% change the color of \emph statements.
```

2.3 Summary of internal macros

The `\Declare...Option` commands define macros, additionally to the macros generated by the key definition. These macros can be used by the package/class

author. The name of the macros starts with the prefix $\langle prefix \rangle$ that can be configured by `\SetupKeyvalOptions`.

Declare $\langle key \rangle$	Defined macro	Description
<code>\DeclareStringOption</code>	<code>\langle prefix \rangle \langle key \rangle</code>	holds the string
<code>\DeclareBoolOption</code>	<code>\if \langle prefix \rangle \langle key \rangle</code> <code>\langle prefix \rangle \langle key \rangle false</code> <code>\langle prefix \rangle \langle key \rangle true</code>	boolean switch disable switch enable switch
<code>\DeclareComplementaryOption</code>	<code>\langle prefix \rangle \langle key \rangle false</code> <code>\langle prefix \rangle \langle key \rangle true</code>	enable parent switch disable parent switch
<code>\DeclareVoidOption</code>	<code>\langle prefix \rangle \langle key \rangle</code>	holds the action

2.4 plain-TeX

Package `keyval` is also usable in plain-TeX with the help of file `miniltx.tex`. Some features of this package `kvoptions` might also be useful for plain-TeX. If \LaTeX is not found, `\ProcessKeyvalOptions` and option `patch` are disabled. Before using the option declaration commands `\Declare...Option`, `\SetupKeyvalOptions` must be used.

3 Example

The following example defined a package that serves some private color management. A boolean option `print` enables print mode without colors. An option `emph` redefines `\emph` to print in the given color. And the driver can be specified by option `driver`.

```

1  $\langle *example \rangle$ 
2   % Package identification
3   % -----
4   \NeedsTeXFormat{LaTeX2e}
5   \ProvidesPackage{example-mycolorsetup}[2009/07/17 Managing my colors]
6
7   \RequirePackage{ifpdf}
8   \RequirePackage{kvoptions}
9
10  % Option declarations
11  % -----
12
13  \SetupKeyvalOptions{
14    family=MCS,
15    prefix=MCS@
16  }
17  % Use a shorter family name and prefix
18
19  % Option print
20  \DeclareBoolOption{print}
21  % is the same as
22  % \DeclareBoolOption[false]{print}
23
24  % Option driver
25  \ifpdf
26    \DeclareStringOption[pdftex]{driver}
27  \else
28    \DeclareStringOption[dvips]{driver}
29  \fi
30
31  % Alternative interface for driver options
32  \DeclareVoidOption{dvips}{\SetupDriver}

```



```

33 \DeclareVoidOption{dvipdfm}{\SetupDriver}
34 \DeclareVoidOption{pdftex}{\SetupDriver}
35 % In \SetupDriver we take the current option \CurrentOption
36 % and pass it to the driver option.
37 % The \expandafter commands expand \CurrentOption at the
38 % time, when \SetupDriver is executed and \CurrentOption
39 % has the correct meaning.
40 \newcommand*{\SetupDriver}{%
41   \expandafter\@SetupDriver\expandafter{\CurrentOption}%
42 }
43 \newcommand*{\@SetupDriver}[1]{%
44   \setkeys{MCS}{driver={#1}}%
45 }
46
47 % Option emph
48 % An empty value means, we want to have no color for \emph.
49 % If the user specifies option emph without value, the red is used.
50 \DeclareStringOption{emph}[red]
51 % is the same as
52 % \DeclareStringOption[]{emph}[red]
53
54 % Default option rule
55 \DeclareDefaultOption{%
56   \ifx\CurrentOptionValue\relax
57     \PackageWarningNoLine{\@currname}{%
58       Unknown option '\CurrentOption'\MessageBreak
59       is passed to package 'color'%
60     }%
61     % Pass the option to package color.
62     % Again it is better to expand \CurrentOption.
63     \expandafter\PassOptionsToPackage
64     \expandafter{\CurrentOption}{color}%
65   \else
66     % Package color does not take options with values.
67     % We provide the standard LaTeX error.
68     \@unknownoptionerror
69   \fi
70 }
71
72 % Process options
73 % -----
74 \ProcessKeyvalOptions*
75
76 % Implementation depending on option values
77 % -----
78 % Code for print mode
79 \ifMCS@print
80   \PassOptionsToPackage{monochrome}{color}
81   % tells package color to use black and white
82 \fi
83
84 \RequirePackage[\MCS@driver]{color}
85 % load package color with the correct driver
86
87 % \emph setup
88 \ifx\MCS@emph\@empty
89   % \@empty is a predefined macro with empty contents.
90   % the option value of option emph is empty, thus
91   % we do not want a redefinition of \emph.
92 \else
93   \renewcommand*{\emph}[1]{%
94     \textcolor{\MCS@emph}{#1}%

```

```

95 }
96 \fi
97 \end{example}

```

4 Package options

The package `kvoptions` knows two package options `patch` and `debugshow`. The options of package `kvoptions` are intended for authors, not for package/class writers. Inside a package it is too late for option `patch` and `debugshow` enables some messages that are perhaps useful for the debugging phase. Also \LaTeX is unhappy if a package is loaded later again with options that are previously not given. Thus package and class authors, stay with `\RequirePackage{kvoptions}` without options.

Option `patch` loads package `kvoptions-patch`.

4.1 Package `kvoptions-patch`

\LaTeX 's system of package/class options has some severe limitations that especially affects the value part if options are used as pair of key and value.

- Spaces are removed, regardless where:

```
\documentclass[box=0 0 400 600]{article}
```

Now each package will see `box=00400600` as global option.

- In the previous case also braces would not help:

```
\documentclass[box={0 0 400 600}]{article}
```

The result is an error message:

```
! LaTeX Error: Missing \begin{document}.
```

As local option, however, it works if the package knows about key value options (By using this package, for example).

- The requirements on robustness are extremely high. \LaTeX expands the option. All that will not work as environment name will break also as option. Even a `\relax` will generate an error message:

```
! Missing \endcsname inserted.
```

Of course, \LaTeX does not use its protecting mechanisms. On contrary `\protect` itself will cause errors.

- The options are expanded. But perhaps the package will do that, because it has to setup some things before? Example `hyperref`:

```
\usepackage[pdfauthor=M\"uller]{hyperref}
```

Package `hyperref` does not see `M\"uller` but its expansion and it does not like it, you get many warnings

```
Token not allowed in a PDFDocEncoded string
```

And the title becomes: `Mu127uller`. Therefore such options must usually be given after package `hyperref` is loaded:

```
\usepackage{hyperref}
\hypersetup{pdfauthor=Fran\c coise M\"uller}
```

As package option it will even break with `Fran\c coise` because of the cedilla `\c`, it is not robust enough.

For users that do not want with this limitations the package offers option `patch`. It patches L^AT_EX's option system and tries to teach it also to handle options that are given as pairs of key and value and to prevent expansion. It can already be used at the very beginning, before `\documentclass`:

```
\RequirePackage[patch]{kvoptions}
\documentclass[pdauthor=Fran\c coise M\"uller]{article}
\usepackage{hyperref}
```

The latest time is before the package where you want to use problematic values:

```
\usepackage[patch]{kvoptions}
\usepackage[Fran\c coise M\"uller]{hyperref}
```

Some remarks:

- The patch requires ε -T_EX, its `\unexpanded` feature is much to nice. It is possible to work around using token registers. But the code becomes longer, slower, more difficult to read and maintain. The package without option `patch` works and will work without ε -T_EX.
- The code for the patch is quite long, there are many test cases. Thus the probability for bugs is probably not too small.

4.2 Option debugshow

The name of this option follows the convention of packages `multicol`, `tabularx`, and `tracefmt`. Currently it prints the setting of boolean options, declared by `\DeclareBoolOption` in the `.log` file, if that boolean option is used. You can activate the option by

- `\PassOptionsToPackage{debugshow}{kvoptions}`
Put this somewhere before package `kvoptions` is loaded first, e.g. before `\documentclass`.
- `\RequirePackage[debugshow]{kvoptions}`
Before `\documentclass` even an author has to use `\RequirePackage`. `\usepackage` only works after `\documentclass`.

The preferred method is `\PassOptionsToPackage`, because it does not force the package loading and does not disturb, if the package is not loaded later at all.

5 Limitations

5.1 Compatibility

5.1.1 Package `kvoptions-patch` vs. package `xkvltxp`

Package `xkvltxp` from the `xkeyval` project has the same goal as package `kvoptions-patch` and to patch L^AT_EX's kernel commands in order to get better support for key value options. Of course they cannot be used both. The user must decide, which method he prefers. Package `kvoptions-patch` aborts itself, if it detects that `xkvltxp` is already loaded.

However package `xkvltxp` and `kvoptions` can be used together, example:

```
\usepackage{xkvltxp}
\usepackage[...]{foobar} % foobar using kvoptions
```

The other way should work, too.

Package `kvoptions-patch` tries to catch more situations and to be more robust. For example, during the comparison of options it normalizes them by removing spaces around `=` and the value. Thus the following is not reported as option clash:

```

\RequirePackage{kvoptions-patch}
\documentclass{article}

\usepackage[scaled=0.7]{helvet}
\usepackage[scaled = 0.7]{helvet}

\begin{document}
\end{document}

```

5.2 Limitations

5.2.1 Option comparisons

In some situations L^AT_EX compares option lists, e.g. option clash check, `\@ifpackagewith`, or `\@ifclasswith`. Apart from catcode and sanitizing problems of option patch, there is another problem. L^AT_EX does not know about the type and default values of options in key value style. Thus an option clash is reported, even if the key value has the same meaning:

```

\usepackage[scaled]{helvet} % default is .95
\usepackage[.95]{helvet}
\usepackage[0.95]{helvet}

```

5.2.2 Option list parsing with package kvoptions-patch

With package kvoptions-patch the range of possible values in key value specifications is much large, for example the comma can be used, if enclosed in curly braces.

Other packages, especially the packages that uses their own process option code can be surprised to find tokens inside options that they do not expect and errors would be the consequence. To avoid errors the options, especially the unused option list is sanitized. That means the list will only contain tokens with catcode 12 (other) and perhaps spaces (catcode 10). This allows a safe parsing for other packages. But a comma in the value part is no longer protected by curly braces because they have lost their special meaning. This is the price for compatibility.

Example:

```

\RequirePackage{kvoptions-patch}
\documentclass[a={a,b,c},b]{article}
\begin{document}
\end{document}

```

Result:

```

LaTeX Warning: Unused global option(s):
[a={a,c},b].

```

6 Implementation

6.1 Preamble

```

98 <*package>

```

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

```

99 \begingroup
100 \catcode44 12 % ,
101 \catcode45 12 % -
102 \catcode46 12 % .
103 \catcode58 12 % :
104 \catcode64 11 % @

```

```

105 \expandafter\let\expandafter\x\csname ver@kvoptions.sty\endcsname
106 \ifcase 0%
107   \ifx\x\relax % plain
108   \else
109     \ifx\x\empty % LaTeX
110     \else
111       1%
112     \fi
113   \fi
114 \else
115   \catcode35 6 % #
116   \catcode123 1 % {
117   \catcode125 2 % }
118   \expandafter\ifx\csname PackageInfo\endcsname\relax
119     \def\x#1#2{%
120       \immediate\write-1{Package #1 Info: #2.}%
121     }%
122   \else
123     \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
124   \fi
125   \x{kvoptions}{The package is already loaded}%
126   \endgroup
127   \expandafter\endinput
128 \fi
129 \endgroup

```

Package identification:

```

130 \begingroup
131   \catcode35 6 % #
132   \catcode40 12 % (
133   \catcode41 12 % )
134   \catcode44 12 % ,
135   \catcode45 12 % -
136   \catcode46 12 % .
137   \catcode47 12 % /
138   \catcode58 12 % :
139   \catcode64 11 % @
140   \catcode123 1 % {
141   \catcode125 2 % }
142   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
143     \def\x#1#2#3[#4]{\endgroup
144       \immediate\write-1{Package: #3 #4}%
145       \xdef#1{#4}%
146     }%
147   \else
148     \def\x#1#2[#3]{\endgroup
149       #2[#{#3}]%
150       \ifx#1\@undefined
151         \xdef#1{#3}%
152       \fi
153       \ifx#1\relax
154         \xdef#1{#3}%
155       \fi
156     }%
157   \fi
158 \expandafter\x\csname ver@kvoptions.sty\endcsname
159 \ProvidesPackage{kvoptions}%
160 [2009/07/17 v3.2 Keyval support for LaTeX options (H0)]

```

Catcodes

```

161 \begingroup
162   \catcode123 1 % {

```

```

163 \catcode125 2 % }
164 \def\x{\endgroup
165   \expandafter\edef\csname KVO@AtEnd\endcsname{%
166     \catcode35 \the\catcode35\relax
167     \catcode64 \the\catcode64\relax
168     \catcode123 \the\catcode123\relax
169     \catcode125 \the\catcode125\relax
170   }%
171 }%
172 \x
173 \catcode35 6 % #
174 \catcode64 11 % @
175 \catcode123 1 % {
176 \catcode125 2 % }
177 \def\TMP@EnsureCode#1#2{%
178   \edef\KVO@AtEnd{%
179     \KVO@AtEnd
180     \catcode#1 \the\catcode#1\relax
181   }%
182   \catcode#1 #2\relax
183 }
184 \TMP@EnsureCode{1}{14}% ^^A (comment)
185 \TMP@EnsureCode{2}{14}% ^^A (comment)
186 \TMP@EnsureCode{33}{12}% !
187 \TMP@EnsureCode{39}{12}% '
188 \TMP@EnsureCode{40}{12}% (
189 \TMP@EnsureCode{41}{12}% )
190 \TMP@EnsureCode{42}{12}% *
191 \TMP@EnsureCode{44}{12}% ,
192 \TMP@EnsureCode{45}{12}% -
193 \TMP@EnsureCode{46}{12}% .
194 \TMP@EnsureCode{47}{12}% /
195 \TMP@EnsureCode{58}{12}% :
196 \TMP@EnsureCode{61}{12}% =
197 \TMP@EnsureCode{62}{12}% >
198 \TMP@EnsureCode{94}{7}% ^ (superscript)
199 \TMP@EnsureCode{96}{12}% ‘

```

External resources. The package extends the support for key value pairs of package `\keyval` to package options. Thus the package needs to be loaded anyway, and we use it for `\SetupKeyvalOptions`. AFAIK this does not disturb users of `xkeyval`.

```

200 \@ifundefined{define@key}{%
201   \RequirePackage{keyval}\relax
202 }{}

```

Provide macros for plain-TeX.

```

203 \@ifundefined{@onelevel@sanitize}{%
204   \def\@onelevel@sanitize#1{%
205     \edef#1{\expandafter\strip@prefix\meaning#1}%
206   }%
207 }{}
208 \@ifundefined{strip@prefix}{%
209   \def\strip@prefix#1>{%
210   }{}
211 \@ifundefined{@x@protect}{%
212   \def\@x@protect#1\fi#2#3{%
213     \fi\protect#1%
214   }%
215   \let\@typeset@protect\relax
216 }{}

```

```

217 \@ifundefined{@currname}{%
218   \def\@currname{}%
219 }{}
220 \@ifundefined{@currentx}{%
221   \def\@currentx{}%
222 }{}

```

Options Option `debugshow` enables additional lines of code that prints information into the `.log` file.

```

223 \DeclareOption{debugshow}{\catcode\@ne=9 }
224 \DeclareOption{patch}{%
225   \AtEndOfPackage{%
226     \RequirePackage{kvoptions-patch}[2009/07/17]%
227   }%
228 }

```

Optionen auswerten:

```

229 \ProcessOptions\relax

```

6.2 Option declaration macros

6.2.1 \SetupKeyvalOptions

The family for the key value pairs can be setup once and is remembered later. The package name seems a reasonable default for the family key, if it is not set by the package author.

`\KV0@family` We cannot store the family setting in one macro, because the package should be usable for many other packages, too. Thus we remember the family setting in a macro, whose name contains the package name with extension, a key in L^AT_EX's class/package system.

```

230 \define@key{KV0}{family}{%
231   \expandafter\edef\csname KV0@family@%
232     \@currname.\@currentx\endcsname{#1}%
233 }
234 \def\KV0@family{%
235   \@ifundefined{KV0@family@\@currname.\@currentx}{%
236     \@currname
237   }{%
238     \csname KV0@family@\@currname.\@currentx\endcsname
239   }%
240 }

```

`\KV0@prefix` The value settings of options that are declared by `\DeclareBoolOption` and `\DeclareStringOption` need to be saved in macros. in the first case this is a switch `\if<prefix><key>`, in the latter case a macro `\<prefix><key>`. The prefix can be configured, by `prefix` that is declared here. The default is the package name with `@` appended.

```

241 \define@key{KV0}{prefix}{%
242   \expandafter\edef\csname KV0@prefix@%
243     \@currname.\@currentx\endcsname{#1}%
244 }
245 \def\KV0@prefix{%
246   \@ifundefined{KV0@prefix@\@currname.\@currentx}{%
247     \@currname @%
248   }{%
249     \csname KV0@prefix@\@currname.\@currentx\endcsname
250   }%
251 }

```

`\SetupKeyvalOptions` The argument of `\SetupKeyvalOptions` expects a key value list, known keys are family and prefix.

```

252 \newcommand*{\SetupKeyvalOptions}{%
253   \setkeys{KVO}%
254 }
```

6.2.2 `\DeclareBoolOption`

`\DeclareBoolOption` Usually options of boolean type can be given by the user without value and this means a setting to *true*. We follow this convention here. Also it simplifies the user interface.

The switch is created and initialized with *false*. The default setting can be overwritten by the optional argument.

L^AT_EX's `\newif` does not check for already defined macros, therefore we add this check here to prevent the user from accidentally redefining of T_EX's primitives and other macros.

```

255 \newcommand*{\DeclareBoolOption}[2][false]{%
256   \KVO@ifdefinable{if\KVO@prefix#2}{%
257     \KVO@ifdefinable{\KVO@prefix#2true}{%
258       \KVO@ifdefinable{\KVO@prefix#2false}{%
259         \csname newif\expandafter\endcsname
260         \csname if\KVO@prefix#2\endcsname
261         \@ifundefined{\KVO@prefix#2#1}{%
262           \PackageWarning{kvoptions}{%
263             Initialization of option ‘#2’ failed,\MessageBreak
264             cannot set boolean option to ‘#1’,\MessageBreak
265             use ‘true’ or ‘false’, now using ‘false’%
266           }%
267         }{%
268           \csname\KVO@prefix#2#1\endcsname
269         }%
270         \begingroup
271         \edef\x{\endgroup
272           \noexpand\define@key{\KVO@family}{#2}[true]{%
273             \noexpand\KVO@boolkey{\@currname}%
274             \ifx\@currentx\@clsextension
275               \noexpand\@clsextension
276             \else
277               \noexpand\@pkgextension
278             \fi
279             {\KVO@prefix}{#2}{####1}%
280           }%
281         }%
282         \x
283       }%
284     }%
285   }%
286 }
```

`\DeclareComplementaryOption` The first argument is the key name, the second the key that must be a boolean option with the same current family and prefix. A new switch is not created for the new key, we have already a switch. Instead we define switch setting commands to work on the parent switch.

```

287 \newcommand*{\DeclareComplementaryOption}[2]{%
288   \@ifundefined{if\KVO@prefix#2}{%
289     \PackageError{kvoptions}{%
290       Cannot generate option code for ‘#1’,\MessageBreak
291       parent switch ‘#2’ does not exist%
292     }{%
293       You are inside %
294       \ifx\@currentx\@clsextension class\else package\fi\space
```



```

295     '@currname.\@current'.\MessageBreak
296     '\KV0@family' is used as familiy %
297     for the keyval options.\MessageBreak
298     '\KV0@prefix' serves as prefix %
299     for internal switch macros.\MessageBreak
300     \MessageBreak
301     \@ehc
302 }%
303 }{%
304     \KV0@ifdefinable{\KV0@prefix#1true}{%
305         \KV0@ifdefinable{\KV0@prefix#1false}{%
306             \expandafter\let\csname\KV0@prefix#1false\expandafter\endcsname
307             \csname\KV0@prefix#2true\endcsname
308             \expandafter\let\csname\KV0@prefix#1true\expandafter\endcsname
309             \csname\KV0@prefix#2false\endcsname

```

The same code part as in \DeclareBoolOption can now be used.

```

310     \begingroup
311     \edef\x{\endgroup
312         \noexpand\define@key{\KV0@family}{#1}[true]{%
313             \noexpand\KV0@boolkey{\@currname}%
314             \ifx\@current\@clsextension
315                 \noexpand\@clsextension
316             \else
317                 \noexpand\@pkgextension
318             \fi
319             {\KV0@prefix}{#1}{####1}%
320         }%
321     }%
322     \x
323 }%
324 }%
325 }%
326 }

```

\KV0@ifdefinable Generate the command token LaTeX's \@ifdefinable expects.

```

327 \def\KV0@ifdefinable#1{%
328     \expandafter\@ifdefinable\csname #1\endcsname
329 }

```

\KV0@boolkey We check explicitly for true and false to prevent the user from accidently calling other macros.

```

#1 package/class name
#2 \@pkgextension/\@clsextension
#3 prefix
#4 key name
#5 new value

```

```

330 \def\KV0@boolkey#1#2#3#4#5{%
331     \edef\KV0@param{#5}%
332     \@onelevel@sanitize\KV0@param
333     \ifx\KV0@param\KV0@true
334         \expandafter\@firstofone
335     \else
336         \ifx\KV0@param\KV0@false
337             \expandafter\expandafter\expandafter\@firstofone
338         \else
339             \ifx#2\@clsextension
340                 \expandafter\ClassWarning
341             \else
342                 \expandafter\PackageWarning
343             \fi

```

```

344     {#1}{%
345         Value '\KV0@param' is not supported by\MessageBreak
346         option '#4'%
347     }%
348     \expandafter\expandafter\expandafter\@gobble
349     \fi
350 \fi
351 {%
352     ^^A\ifx#2\@clsextension
353     ^^A \expandafter\ClassInfo
354     ^^A\else
355     ^^A \expandafter\PackageInfo
356     ^^A\fi
357     ^^A{#1}{[option] #4=\KV0@param}%
358     \csname#3#4\KV0@param\endcsname
359 }%
360 }

```

\KV0@true The macros \KV0@true and \KV0@false are used for string comparisons. After
 \KV0@false \@onelevel@sanitize we have only tokens with catcode 12 (other).

```

361 \def\KV0@true{true}
362 \def\KV0@false{false}
363 \@onelevel@sanitize\KV0@true
364 \@onelevel@sanitize\KV0@false

```

6.2.3 \DeclareStringOption

\DeclareStringOption

```

365 \newcommand*{\DeclareStringOption}[2][]{%
366     \@ifnextchar[%
367         \KV0@DeclareStringOption{#1}{#2}@%
368     ]{%
369         \KV0@DeclareStringOption{#1}{#2}{}%
370     }%
371 }

```

\KV0@DeclareStringOption

```

372 \def\KV0@DeclareStringOption#1#2#3[#4]{%
373     \KV0@ifdefinable{\KV0@prefix#2}{%
374         \@namedef{\KV0@prefix#2}{#1}%
375         \begingroup
376             \ifx\#3\%
377                 \toks@{%
378                 \else
379                     \toks@{[#4]}%
380                 \fi
381                 \edef\x{\endgroup
382                     \noexpand\define@key{\KV0@family}{#2}\the\toks@{%
383                     ^^A\begingroup
384                     ^^A \toks@{####1}%
385                     ^^A \ifx\@current\@clsextension
386                     ^^A \noexpand\ClassInfo
387                     ^^A \else
388                     ^^A \noexpand\PackageInfo
389                     ^^A \fi
390                     ^^A {\@currname}{%
391                     ^^A [option] #2={\noexpand\the\toks@}%
392                     ^^A }%
393                     ^^A\endgroup
394                     \noexpand\def
395                     \expandafter\noexpand\csname\KV0@prefix#2\endcsname{####1}%
396                 }%

```

```

397     }%
398     \x
399 }%
400 }

```

6.2.4 \DeclareVoidOption

\DeclareVoidOption

```

401 \newcommand*{\DeclareVoidOption}[1]{%
402   \begingroup
403     \let\next\@gobbletwo
404     \KV0@ifdefinable{\KV0@prefix#1}{%
405       \let\next\@firstofone
406     }%
407   \expandafter\endgroup
408   \next{%
409     \begingroup
410     \edef\x{\endgroup
411       \noexpand\define@key{\KV0@family}{#1}[\KV0@VOID@]{%
412         \noexpand\KV0@voidkey{\@currname}%
413         \ifx\@current\@clsextension
414           \noexpand\@clsextension
415         \else
416           \noexpand\@pkgextension
417         \fi
418         {#1}%
419         {###1}%
420         \expandafter\noexpand\csname\KV0@prefix#1\endcsname
421       }%
422     }%
423     \x
424     \@namedef{\KV0@prefix#1}%
425   }%
426 }
427 \def\KV0@VOID@{@VOID@}

```

#1 package/class name
 #2 \@pkgextension/\@clsextension
 \KV0@voidkey #3 key name
 #4 default (@VOID@)
 #5 macro with option code

```

428 \def\KV0@voidkey#1#2#3#4{%
429   \def\CurrentOption{#3}%
430   \begingroup
431     \def\x{#4}%
432   \expandafter\endgroup
433   \ifx\x\KV0@VOID@
434   \else
435     \ifx#2\@clsextension
436       \expandafter\ClassWarning
437     \else
438       \expandafter\PackageWarning
439     \fi
440     {#1}{%
441       Unexpected value for option ‘#3’\MessageBreak
442       is ignored%
443     }%
444   \fi
445   ^^A\ifx#2\@clsextension
446   ^^A \expandafter\ClassInfo
447   ^^A\else
448   ^^A \expandafter\PackageInfo

```

```

449 ^^A\fi
450 ^^A{#1}{[option] #3}%
451 }

```

6.2.5 \DeclareDefaultOption

\DeclareDefaultOption

```

452 \newcommand*{\DeclareDefaultOption}{%
453   \@namedef{KV0@default@ \@currname.\@current}%
454 }

```

6.3 Dynamic options

6.3.1 \DisableKeyvalOption

```

455 \SetupKeyvalOptions{%
456   family=KV0dyn,%
457   prefix=KV0dyn@%
458 }
459 \DeclareBoolOption[true]{global}
460 \DeclareComplementaryOption{local}{global}
461 \DeclareStringOption[undef]{action}
462 \let\KV0dyn@name\relax
463 \let\KV0dyn@ext\@empty
464 \define@key{KV0dyn}{class}{%
465   \def\KV0dyn@name{#1}%
466   \let\KV0dyn@ext\@clsextension
467 }
468 \define@key{KV0dyn}{package}{%
469   \def\KV0dyn@name{#1}%
470   \let\KV0dyn@ext\@pkgextension
471 }
472 \newcommand*{\DisableKeyvalOption}[3][{}]{%
473   \begingroup
474     \setkeys{KV0dyn}{#1}%
475     \def\x{\endgroup}%
476     \@ifundefined{KV0@action@KV0dyn@action}{%
477       \PackageError{kvoptions}{%
478         Unknown disable action %
479         '\expandafter\strip@prefix\meaning\KV0dyn@action'\MessageBreak
480         for option '#3' in keyval family '#2'%
481       }\@ehc
482     }{%
483       \csname KV0@action@KV0dyn@action\endcsname{#2}{#3}%
484     }%
485   \x
486 }
487 \def\KV0@action@undef#1#2{%
488   \edef\x{\endgroup
489     \ifKV0dyn@global\global\fi
490     \let
491     \expandafter\noexpand\csname KV@#1@#2\endcsname
492     \relax
493     \ifKV0dyn@global\global\fi
494     \let
495     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
496     \relax
497   }%
498   ^^A\PackageInfo{kvoptions}{%
499     ^^A [option] key '#2' of family '#1'\MessageBreak
500     ^^A is disabled (undef, \ifKV0dyn@global\global\else local\fi)%
501   ^^A}%

```

```

502 }
503 \def\KV0@action@ignore#1#2{%
504   \edef\x{\endgroup
505     \ifKV0dyn@global\global\fi
506     \let
507     \expandafter\noexpand\csname KV@#1@#2\endcsname
508     \noexpand@gobble
509     \ifKV0dyn@global\global\fi
510     \let
511     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
512     \noexpand@empty
513   }%
514   ^^A\PackageInfo{kvoptions}{%
515     ^^A [option] key ‘#2’ of family ‘#1’\MessageBreak
516     ^^A is disabled (ignore, \ifKV0dyn@global\global\else local\fi)%
517   ^^A}%
518 }
519 \def\KV0@action@error{%
520   \KV0@do@action{error}%
521 }
522 \def\KV0@action@warning{%
523   \KV0@do@action{warning}%
524 }

#1 error or warning
#2 <family>
#3 <key>
525 \def\KV0@do@action#1#2#3{%
526   \ifx\KV0dyn@name\relax
527     \PackageError{kvoptions}{%
528       Action type ‘#1’ needs package/class name\MessageBreak
529       for key ‘#3’ in family ‘#2’%
530     }\@ehc
531   \else
532     \edef\x{\endgroup
533       \noexpand\define@key{#2}{#3}[]{%
534         \expandafter\noexpand\csname KV0@disable@#1\endcsname
535         {\KV0dyn@name}\noexpand\KV0dyn@ext{#3}%
536       }%
537       \ifKV0dyn@global
538         \global\let
539         \expandafter\noexpand\csname KV@#2@#3\endcsname
540         \expandafter\noexpand\csname KV@#2@#3\endcsname
541         \global\let
542         \expandafter\noexpand\csname KV@#2@#3@default\endcsname
543         \expandafter\noexpand\csname KV@#2@#3@default\endcsname
544       \fi
545     }%
546     ^^A\ifx\KV0dyn@ext\@clsextension
547     ^^A \expandafter\ClassInfo
548     ^^A\else
549     ^^A \expandafter\PackageInfo
550     ^^A\fi
551     ^^A{\KV0dyn@name}{%
552     ^^A [option] key ‘#3’ of family ‘#2’\MessageBreak
553     ^^A is disabled (#1, \ifKV0dyn@global\global\else local\fi)%
554     ^^A}%
555   \fi
556 }
557 \def\KV0@disable@error#1#2#3{%
558   \ifx#2\@clsextension
559     \expandafter\ClassError
560   \else

```

```

561     \expandafter\PackageError
562 \fi
563 {#1}{%
564     Option ‘#3’ is given too late,\MessageBreak
565     now the option is ignored%
566 }\@ehc
567 }
568 \def\KV0@disable@warning#1#2#3{%
569     \ifx#2\@clsextension
570         \expandafter\ClassWarning
571     \else
572         \expandafter\PackageWarning
573     \fi
574 {#1}{%
575     Option ‘#3’ is already consumed\MessageBreak
576     and has no effect%
577 }%
578 }

```

6.4 Process options

6.5 \ProcessKeyvalOptions

`\ProcessKeyvalOptions` If the optional star is given, we get the family name and expand it for safety.

```

579 \newcommand*{\ProcessKeyvalOptions}{%
580     \@ifstar{%
581         \begingroup
582         \edef\x{\endgroup
583             \noexpand\KV0@ProcessKeyvalOptions{\KV0@family}%
584         }%
585         \x
586     }%
587     \KV0@ProcessKeyvalOptions
588 }

589 \def\KV0@ProcessKeyvalOptions#1{%
590     \let\@tempc\relax
591     \let\KV0@temp\@empty

```

Add any global options that are known to KV to the start of the list being built in `\KV0@temp` and mark them used (by removing them from the unused option list).

```

592     \ifx\@currentx\@clsextension
593     \else
594         \ifx\@classoptionslist\relax
595         \else
596             \@for\KV0@CurrentOption:=\@classoptionslist\do{%
597                 \@ifundefined{KV@#1}\expandafter\KV0@getkey
598                     \KV0@CurrentOption=\@nil}{%
599                 }{%
600                     \ifx\KV0@Patch Y%
601                         \edef\KV0@temp{%
602                             \etex@unexpanded\expandafter{%
603                                 \KV0@temp
604                             }%
605                             ,%
606                             \etex@unexpanded\expandafter{%
607                                 \KV0@CurrentOption
608                             }%
609                             ,%
610                         }%
611                         \@onelevel@sanitize\KV0@CurrentOption
612                     \else
613                         \edef\KV0@temp{%
614                             \KV0@temp

```

```

615         ,%
616         \KV0@CurrentOption
617         ,%
618     }%
619 \fi
620 \expandafter\@removeelement\KV0@CurrentOption
621 \expandafter\@unusedoptionlist\@unusedoptionlist
622 }%
623 }%
624 \fi
625 \fi

```

Now stick the package options at the end of the list and wrap in a call to `\setkeys`. A class ignores unknown global options, we must remove them to prevent error messages from `\setkeys`.

```

626 \begingroup
627 \toks\tw@{}%
628 \ifundefined{opt@\@currname.\@currentx}{%
629 \toks@{\expandafter{\KV0@temp}}%
630 }{%
631 \toks@{\expandafter\expandafter\expandafter{%
632 \csname opt@\@currname.\@currentx\endcsname
633 }%
634 \ifx\@currentx\@clsextension
635 \edef\CurrentOption{\the\toks@}%
636 \toks@{\expandafter{\KV0@temp}}%
637 \@for\CurrentOption:=\CurrentOption\do{%
638 \ifundefined{%
639 KV@#1\expandafter\KV0@getkey\CurrentOption=\@nil
640 }{%

```

A class puts not used options in the unused option list.

```

641 \ifx\KV0@Patch Y%
642 \onelevel@sanitize\CurrentOption
643 \fi
644 \ifx\@unusedoptionlist\@empty
645 \global\let\@unusedoptionlist\CurrentOption
646 \else
647 \expandafter\expandafter\expandafter\gdef
648 \expandafter\expandafter\expandafter\@unusedoptionlist
649 \expandafter\expandafter\expandafter{%
650 \expandafter\@unusedoptionlist
651 \expandafter,\CurrentOption
652 }%
653 \fi
654 }{%
655 \toks@{\expandafter{%
656 \the\expandafter\toks@\expandafter,\CurrentOption
657 }%
658 }%
659 }%
660 \else

```

Without default action we pass all options to `\setkeys`. Otherwise we have to check which options are known. These are passed to `\setkeys`. For the others the default action is performed.

```

661 \ifundefined{KV0@default@\@currname.\@currentx}{%
662 \toks@{\expandafter\expandafter\expandafter{%
663 \expandafter\KV0@temp\the\toks@
664 }%
665 }{%
666 \edef\CurrentOption{\the\toks@}%
667 \toks@{\expandafter{\KV0@temp}}%
668 \@for\CurrentOption:=\CurrentOption\do{%

```

```

669         \ifundefined{%
670             KV@#1@\expandafter\KV@getkey\CurrentOption=\@nil
671         }{%
672             \toks\tw@\expandafter{%
673                 \the\toks\expandafter\tw@\expandafter,\CurrentOption
674             }%
675         }{%
676             \toks@\expandafter{%
677                 \the\expandafter\toks@\expandafter,\CurrentOption
678             }%
679         }%
680     }%
681 }%
682 \fi
683 }%
684 \edef\KV@temp{\endgroup
685     \noexpand\KV@calldefault{\the\toks\tw@}%
686     \noexpand\setkeys{#1}{\the\toks@}%
687 }%
688 \KV@temp

```

Some cleanup of \ProcessOptions.

```

689 \let\CurrentOption\@empty
690 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
691 }

```

6.6 \ProcessLocalKeyvalOptions

\ProcessLocalKeyvalOptions If the optional star is given, we get the family name and expand it for safety.

```

692 \newcommand*{\ProcessLocalKeyvalOptions}{%
693     \@ifstar{%
694         \begingroup
695         \edef\x{\endgroup
696             \noexpand\KV@ProcessLocalKeyvalOptions{\KV@family}%
697         }%
698         \x
699     }%
700     \KV@ProcessLocalKeyvalOptions
701 }

702 \def\KV@ProcessLocalKeyvalOptions#1{%
703     \let\@tempc\relax
704     \let\KV@temp\@empty

```

Check if \ProcessLocalKeyvalOptions is called inside a package.

```

705     \ifx\@current\@pkgextension
706     \else
707         \PackageError{kvoptions}{%
708             \string\ProcessLocalKeyvalOptions is intended for packages only%
709         }\@ehc
710     \fi

```

The package options are put into toks register \toks@.

```

711     \begingroup
712     \toks\tw@{}%
713     \ifundefined{opt@\@currname.\@current}{%
714         \toks@\expandafter{\KV@temp}%
715     }{%
716         \toks@\expandafter\expandafter\expandafter{%
717             \csname opt@\@currname.\@current\endcsname
718         }%

```


Without default action we pass all options to `\setkeys`. Otherwise we have to check which options are known. These are passed to `\setkeys`. For the others the default action is performed.

```

719      \@ifundefined{KV0@default@\currname.\@currentx}{%
720        \toks@\expandafter\expandafter\expandafter{%
721          \expandafter\KV0@temp\the\toks@
722        }%
723      }{%
724        \edef\CurrentOption{\the\toks@}%
725        \toks@\expandafter\KV0@temp}%
726        \@for\CurrentOption:=\CurrentOption\do{%
727          \@ifundefined{%
728            KV@#1\expandafter\KV0@getkey\CurrentOption=\@nil
729          }{%
730            \toks\tw@\expandafter{%
731              \the\toks\expandafter\tw@\expandafter,\CurrentOption
732            }%
733          }{%
734            \toks@\expandafter{%
735              \the\expandafter\toks@\expandafter,\CurrentOption
736            }%
737          }%
738        }%
739      }%
740    \fi
741  }%
742  \edef\KV0@temp{\endgroup
743    \noexpand\KV0@calldefault{\the\toks\tw@}%
744    \noexpand\setkeys{#1}{\the\toks@}%
745  }%
746  \KV0@temp

```

Some cleanup of `\ProcessOptions`.

```

747  \let\CurrentOption\@empty
748  \AtEndOfPackage{\let\@unprocessedoptions\relax}%
749 }

```

6.6.1 Helper macros

`\KV0@getkey` Extract the key part of a key=value pair.

```

750 \def\KV0@getkey#1=#2\@nil{#1}

```

`\KV0@calldefault`

```

751 \def\KV0@calldefault#1{%
752   \begingroup
753   \def\x{#1}%
754   \expandafter\endgroup
755   \ifx\x\@empty
756   \else
757     \@for\CurrentOption:=#1\do{%
758       \ifx\CurrentOption\@empty
759       \else
760         \expandafter\KV0@setcurrents\CurrentOption=\@nil
761         \@nameuse{KV0@default@\currname.\@currentx}%
762       \fi
763     }%
764   \fi
765 }

```

`\KV0@setcurrents` Extract the key part of a key=value pair.

```

766 \def\KV0@setcurrents#1=#2\@nil{%
767   \def\CurrentOptionValue{#2}%

```

```

768 \ifx\CurrentOptionValue\@empty
769   \let\CurrentOptionKey\CurrentOption
770   \let\CurrentOptionValue\relax
771 \else
772   \edef\CurrentOptionKey{\zap@space#1 \@empty}%
773   \expandafter\KV@setcurrentvalue\CurrentOption\@nil
774 \fi
775 }

```

`\KV@setcurrentvalue` Here the value part is parsed. Package `keyval`'s `\KV@@sp@def` helps in removing spaces at the begin and end of the value.

```

776 \def\KV@setcurrentvalue#1=#2\@nil{%
777   \KV@@sp@def\CurrentOptionValue{#2}%
778 }

```

6.7 plain-TeX

Disable L^AT_EX stuff.

```

779 \begingroup\expandafter\expandafter\expandafter\endgroup
780 \expandafter\ifx\curname documentclass\endcurname\relax
781   \def\ProcessKeyvalOptions{%
782     \@ifstar{}\@gobble
783   }%
784 \fi

785 \KV@AtEnd
786 \</package>

```

6.8 Package `kvoptions-patch`

```

787 \<*patch>
788 \NeedsTeXFormat{LaTeX2e}
789 \begingroup
790   \catcode123 1 % {
791   \catcode125 2 % }
792   \def\x{\endgroup
793     \expandafter\edef\curname KV@AtEnd\endcurname{%
794       \catcode35 \the\catcode35\relax
795       \catcode64 \the\catcode64\relax
796       \catcode123 \the\catcode123\relax
797       \catcode125 \the\catcode125\relax
798     }%
799   }%
800   \x
801   \catcode35 6 % #
802   \catcode64 11 % @
803   \catcode123 1 % {
804   \catcode125 2 % }
805   \def\TMP@EnsureCode#1#2{%
806     \edef\KV@AtEnd{%
807       \KV@AtEnd
808       \catcode#1 \the\catcode#1\relax
809     }%
810     \catcode#1 #2\relax
811   }
812   \TMP@EnsureCode{39}{12}% '
813   \TMP@EnsureCode{40}{12}% (
814   \TMP@EnsureCode{41}{12}% )
815   \TMP@EnsureCode{43}{12}% +
816   \TMP@EnsureCode{44}{12}% ,
817   \TMP@EnsureCode{45}{12}% -
818   \TMP@EnsureCode{46}{12}% .

```

```

819 \TMP@EnsureCode{47}{12}% /
820 \TMP@EnsureCode{58}{12}% :
821 \TMP@EnsureCode{60}{12}% <
822 \TMP@EnsureCode{61}{12}% =
823 \TMP@EnsureCode{62}{12}% >
824 \TMP@EnsureCode{91}{12}% [
825 \TMP@EnsureCode{93}{12}% ]
826 \TMP@EnsureCode{96}{12}% '
827 \TMP@EnsureCode{124}{12}% |
828 \edef\KVO@AtEnd{%
829   \KVO@AtEnd
830   \noexpand\endinput
831 }
832 \ProvidesPackage{kvoptions-patch}%
833 [2009/07/17 v3.2 LaTeX patch for keyval options (HO)]%

Check for  $\varepsilon$ -TeX.
834 \begingroup\expandafter\expandafter\expandafter\endgroup
835 \expandafter\ifx\csname eTeXversion\endcsname\relax
836   \PackageWarningNoLine{kvoptions-patch}{%
837     Package loading is aborted, because e-TeX is missing%
838   }%
839   \expandafter\KVO@AtEnd
840 \fi

Package etexcmds for \etex@unexpanded.
841 \RequirePackage{etexcmds}[2007/09/09]
842 \ifetex@unexpanded
843 \else
844   \PackageError{kvoptions-patch}{%
845     Could not find eTeX's \string\unexpanded.\MessageBreak
846     Try adding \string\RequirePackage\string{etexcmds}\string}%
847     before \string\documentclass%
848   }\@ehd
849   \expandafter\KVO@AtEnd
850 \fi

Check for package xkvltxp.
851 \@ifpackageloaded{xkvltxp}{%
852   \PackageWarningNoLine{kvoptions}{%
853     Option 'patch' cannot be used together with\MessageBreak
854     package 'xkvltxp' that is already loaded.\MessageBreak
855     Therefore package loading is aborted%
856   }%
857   \KVO@AtEnd
858 }{}

859 \def\@if@options#1#2#3{%
860   \begingroup
861     \KVO@normalize\KVO@temp{#3}%
862     \edef\x{\endgroup
863       \noexpand\@if@ptions{%
864         \detokenize\expandafter\expandafter\expandafter{%
865           \csname opt#2.#1\endcsname
866         }%
867       }{%
868         \detokenize\expandafter{\KVO@temp}%
869       }%
870     }%
871   \x
872 }

873 \def\@pass@options#1#2#3{%
874   \KVO@normalize\KVO@temp{#2}%
875   \@ifundefined{opt@#3.#1}{%
876     \expandafter\gdef\csname opt@#3.#1%

```

```

877         \expandafter\endcsname\expandafter{%
878         \KV0@temp
879     }%
880 }{%
881     \expandafter\gdef\csname opt@#3.#1%
882         \expandafter\expandafter\expandafter\endcsname
883         \expandafter\expandafter\expandafter{%
884         \csname opt@#3.#1\expandafter\endcsname\expandafter,\KV0@temp
885     }%
886 }%
887 }
888 \def\ProcessOptions{%
889     \let\ds@\empty
890     \ifundefined{opt@\currname.\@currentx}{%
891         \let\@curroptions\empty
892     }{%
893         \expandafter\expandafter\expandafter\def
894         \expandafter\expandafter\expandafter\@curroptions
895         \expandafter\expandafter\expandafter{%
896         \csname opt@\currname.\@currentx\endcsname
897     }%
898 }%
899 \ifstar\KV0@xprocess@options\KV0@process@options
900 }
901 \def\KV0@process@options{%
902     \for\CurrentOption:=\@declaredoptions\do{%
903         \ifx\CurrentOption\empty
904             \else
905                 \begingroup
906                 \ifx\@currentx\@clsextension
907                     \toks@{}%
908                 \else
909                     \toks@\expandafter{\@classoptionslist,%
910                 \fi
911                 \toks\tw@\expandafter{\@curroptions}%
912                 \edef\x{\endgroup
913                 \noexpand\in@{\CurrentOption,}{,\the\toks@\the\toks\tw@,%}%
914                 }%
915                 \x
916                 \ifin@
917                     \KV0@use@option
918                     \expandafter\let\csname ds@\CurrentOption\endcsname\empty
919                 \fi
920             \fi
921         }%
922     \KV0@process@ptions
923 }
924 \def\KV0@xprocess@options{%
925     \ifx\@currentx\@clsextension
926     \else
927         \for\CurrentOption:=\@classoptionslist\do{%
928             \ifx\CurrentOption\empty
929             \else
930                 \KV0@in@\CurrentOption\@declaredoptions
931                 \ifin@
932                     \KV0@use@option
933                     \expandafter\let\csname ds@\CurrentOption\endcsname\empty
934                 \fi
935             \fi
936         }%
937     \fi
938     \KV0@process@ptions

```

```

939 }
940 \def\KVO@in@#1#2{%
941   \in@false
942   \beginingroup
943     \@for\x:=#2\do{%
944       \ifx\x#1\relax
945         \in@true
946       \fi
947     }%
948   \edef\x{\endgroup
949     \ifin@
950       \noexpand\in@true
951     \fi
952   }%
953   \x
954 }

955 \def\KVO@process@pti@ns{%
956   \@for\CurrentOption:=\@curroptions\do{%
957     \ifundefined{ds@\KVO@SanitizedCurrentOption}{%
958       \KVO@use@option
959       \default@ds
960     }%
961     \KVO@use@option
962   }%
963   \@for\CurrentOption:=\@declaredoptions\do{%
964     \expandafter\let\csname ds@\CurrentOption\endcsname\relax
965   }%
966   \let\CurrentOption@empty
967   \let\@fileswith@pti@ns\@fileswith@pti@ns
968   \AtEndOfPackage{\let\@unprocessedoptions\relax}%
969 }

970 \def\KVO@use@option{%
971   \beginingroup
972     \edef\x{\endgroup
973       \noexpand\@removeelement{%
974         \detokenize\expandafter{\CurrentOption}%
975       }{%
976         \detokenize\expandafter{\@unusedoptionlist}%
977       }%
978     }%
979     \x\@unusedoptionlist
980     \csname ds@\KVO@SanitizedCurrentOption\endcsname
981   }

982 \def\OptionNotUsed{%
983   \ifx\@current\@clsextension
984     \xdef\@unusedoptionlist{%
985       \ifx\@unusedoptionlist\@empty
986         \else
987           \detokenize\expandafter{\@unusedoptionlist},}%
988       \fi
989       \detokenize\expandafter{\CurrentOption}%
990     }%
991   \fi
992 }

Variant of \ExecuteOptions that better protects \CurrentOption.
993 \def\CurrentOption@SaveLevel{0}
994 \def\ExecuteOptions{%
995   \expandafter\KVO@ExecuteOptions
996   \csname CurrentOption@\CurrentOption@SaveLevel\endcsname
997 }
998 \def\KVO@ExecuteOptions#1#2{%

```

```

999 \let#1\CurrentOption
1000 \edef\CurrentOption@SaveLevel{%
1001   \the\numexpr\CurrentOption@SaveLevel+1%
1002 }%
1003 \@for\CurrentOption:=#2\do{%
1004   \csname ds@\CurrentOption\endcsname
1005 }%
1006 \edef\CurrentOption@SaveLevel{%
1007   \the\numexpr\CurrentOption@SaveLevel-1%
1008 }%
1009 \let\CurrentOption#1%
1010 }

1011 \def\KVO@fileswith@ptions#1[#2]#3[#4]{%
1012   \ifx#1\@clsextension
1013     \ifx\@classoptionslist\relax
1014       \KVO@normalize\KVO@temp{#2}%
1015       \expandafter\gdef\expandafter\@classoptionslist\expandafter{%
1016         \KVO@temp
1017       }%
1018       \def\reserved@a{%
1019         \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
1020         \@documentclasshook
1021       }%
1022     \else
1023       \def\reserved@a{%
1024         \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
1025       }%
1026     \fi
1027   \else
1028     \begingroup
1029     \let\KVO@temp\relax
1030     \let\KVO@onefilewithoptions\relax
1031     \let\@pkgextension\relax
1032     \def\reserved@b##1,{%
1033       \ifx\@nil##1\relax
1034       \else
1035         \ifx\relax##1\relax
1036         \else
1037           \KVO@onefilewithoptions{##1}[{\KVO@temp}][#{#4}]%
1038           \@pkgextension
1039         \fi
1040         \expandafter\reserved@b
1041       \fi
1042     }%
1043     \edef\reserved@a{\zap@space#3 \@empty}%
1044     \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
1045     \toks@{#2}%
1046     \def\KVO@temp{\the\toks@}%
1047     \edef\reserved@a{\endgroup \reserved@a}%
1048   \fi
1049   \reserved@a
1050 }

1051 \def\KVO@onefilewithoptions#1[#2]#3[#4]{%
1052   \@pushfilename
1053   \xdef\@currname{#1}%
1054   \global\let\@currentx#4%
1055   \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname\@empty
1056   \let\CurrentOption\@empty
1057   \@reset@ptions
1058   \makeatletter
1059   \def\reserved@a{%
1060     \ifl@aded\@currentx{#1}{%

```

```

1061 \if@options\@currentx{#1}{#2}{%
1062 }{%
1063 \begingroup
1064 \ifundefined{opt@#1.\@currentx}{%
1065 \def\x{%
1066 }{%
1067 \edef\x{%
1068 \expandafter\expandafter\expandafter\strip@prefix
1069 \expandafter\meaning\csname opt@#1.\@currentx\endcsname
1070 }%
1071 }%
1072 \def\y{#2}%
1073 \edef\y{\expandafter\strip@prefix\meaning\y}%
1074 \@latex@error{Option clash for \cls@pkg\space #1}{%
1075 The package #1 has already been loaded %
1076 with options:\MessageBreak
1077 \space\space[\x]\MessageBreak
1078 There has now been an attempt to load it %
1079 with options:\MessageBreak
1080 \space\space[\y]\MessageBreak
1081 Adding the global options:\MessageBreak
1082 \space\space
1083 \x,\y\MessageBreak
1084 to your \noexpand\documentclass declaration may fix this.%
1085 \MessageBreak
1086 Try typing \space <return> \space to proceed.%
1087 }%
1088 \endgroup
1089 }%
1090 }{%
1091 \@pass@options\@currentx{#2}{#1}%
1092 \global\expandafter
1093 \let\csname ver@\@currname.\@currentx\endcsname\@empty
1094 \InputIfFileExists
1095 {\@currname.\@currentx}%
1096 {}%
1097 {\@missingfileerror\@currname\@currentx}%
1098 \let\@unprocessedoptions\@unprocessedoptions
1099 \csname\@currname.\@currentx-h@@k\endcsname
1100 \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname
1101 \undefined
1102 \@unprocessedoptions
1103 }%
1104 \@ifl@ter\@currentx{#1}{#3}{%
1105 }{%
1106 \@latex@warning@no@line{%
1107 You have requested,\on@line, %
1108 version\MessageBreak
1109 #3' of \cls@pkg\space #1,\MessageBreak
1110 but only version\MessageBreak
1111 '\csname ver@#1.\@currentx\endcsname'\MessageBreak
1112 is available%
1113 }%
1114 }%
1115 \ifx\@currentx\cls@extension\let\LoadClass\@twoloadclasserror\fi
1116 \popfilename
1117 \@reset@options
1118 }%
1119 \reserved@a
1120 }

1121 \def\@unknownoptionerror{%
1122 \@latex@error{%

```

```

1123     Unknown option '\KV0@SanitizedCurrentOption' %
1124     for \@cls@pkg\space'\@currname'%
1125 }{%
1126     The option '\KV0@SanitizedCurrentOption' was not declared in %
1127     \@cls@pkg\space'\@currname', perhaps you\MessageBreak
1128     misspelled its name. %
1129     Try typing \space <return> %
1130     \space to proceed.%
1131 }%
1132 }

1133 \def\@@unprocessedoptions{%
1134     \ifx\@current\@pkgextension
1135         \@ifundefined{opt@\@currname.\@current}{%
1136             \let\@curroptions\@empty
1137         }{%
1138             \expandafter\let\expandafter\@curroptions
1139                 \csname opt@\@currname.\@current\endcsname
1140         }%
1141         \@for\CurrentOption:=\@curroptions\do{%
1142             \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
1143         }%
1144     \fi
1145 }

1146 \def\KV0@SanitizedCurrentOption{%
1147     \expandafter\strip@prefix\meaning\CurrentOption
1148 }

    Normalize option list.
1149 \def\KV0@normalize#1#2{%
1150     \let\KV0@result\@empty
1151     \KV0@splitcomma#2,\@nil
1152     \let#1\KV0@result
1153 }
1154 \def\KV0@splitcomma#1,#2\@nil{%
1155     \KV0@ifempty{#1}{}%
1156     \KV0@checkkv#1=\@nil
1157 }%
1158 \KV0@ifempty{#2}{}\KV0@splitcomma#2\@nil}%
1159 }
1160 \def\KV0@ifempty#1{%
1161     \expandafter\ifx\expandafter\\detokenize{#1}\\%
1162         \expandafter\@firstoftwo
1163     \else
1164         \expandafter\@secondoftwo
1165     \fi
1166 }
1167 \def\KV0@checkkv#1=#2\@nil{%
1168     \KV0@ifempty{#2}{%
1169         % option without value
1170         \edef\KV0@x{\zap@space#1 \@empty}%
1171         \ifx\KV0@x\@empty
1172             % ignore empty option
1173         \else
1174             % append to list
1175             \edef\KV0@result{%
1176                 \etex@unexpanded\expandafter{\KV0@result},\KV0@x
1177             }%
1178         \fi
1179     }{%
1180         % #1: "key", #2: "value="
1181         % add key part
1182         \edef\KV0@result{%

```



```

1183     \etex@unexpanded\expandafter{\KV0@result},%
1184     \zap@space#1 \@empty
1185 }%
1186 \futurelet\@let@token\KV0@checkfirsttok#2 \@nil| = \@nil|\KV0@nil
1187 }%
1188 }
1189 \def\KV0@checkfirsttok{%
1190   \ifx\@let@token\bgroup
1191     % no space at start
1192     \expandafter\KV0@removelastspace\expandafter=%
1193     % "<value><spaceopt>= \@nil"
1194   \else
1195     \expandafter\KV0@checkfirstA
1196   \fi
1197 }
1198 \def\KV0@checkfirstA#1 #2\@nil{%
1199   \KV0@ifempty{#2}{%
1200     \KV0@removelastspace=#1 \@nil
1201   }{%
1202     \KV0@ifempty{#1}{%
1203       \KV0@removelastspace=#2\@nil
1204     }{%
1205       \KV0@removelastspace=#1 #2\@nil
1206     }%
1207   }%
1208 }
1209 \def\KV0@removelastspace#1 = \@nil|#2\KV0@nil{%
1210   \KV0@ifempty{#2}{%
1211     \edef\KV0@result{%
1212       \etex@unexpanded\expandafter{\KV0@result}%
1213       \etex@unexpanded\expandafter{\KV0@removegarbage#1\KV0@nil}%
1214     }%
1215   }{%
1216     \edef\KV0@result{%
1217       \etex@unexpanded\expandafter{\KV0@result}%
1218       \etex@unexpanded{#1}%
1219     }%
1220   }%
1221 }
1222 \def\KV0@removegarbage#1= \@nil#2\KV0@nil{#1}%

```

Arguments #1 and #2 are macros.

```

1223 \def\KV0@removeelement#1#2{%
1224   \begingroup
1225     \toks@={}%
1226     \@for\x:=#2\do{%
1227       \ifx\x\@empty
1228       \else
1229         \ifx\x#1\relax
1230         \else
1231           \edef\t{\the\toks@}%
1232           \ifx\t\@empty
1233           \else
1234             \toks@\expandafter{\the\toks@,%}%
1235           \fi
1236           \toks@\expandafter{\the\expandafter\toks@\x}%
1237         \fi
1238       \fi
1239     }%
1240     \edef\x{\endgroup
1241       \def\noexpand#2{\the\toks@}%
1242     }%
1243   \x

```

```

1244 }
1245 \let\@@fileswith@pti@ns\KV0@fileswith@pti@ns
1246 \ifx\@fileswith@pti@ns\@badrequireerror
1247 \else
1248   \let\@fileswith@pti@ns\KV0@fileswith@pti@ns
1249 \fi

\KV0@Patch

1250 \let\KV0@Patch=Y

1251 \KV0@AtEnd
1252 </patch>

```

7 Test

7.1 Preface for standard catcode check

```

1253 <*test1>
1254 \input miniltx.tex\relax
1255 </test1>

```

7.2 Catcode checks for loading

```

1256 <*test1>

1257 \catcode'\{=1 %
1258 \catcode'\}=2 %
1259 \catcode'\#=6 %
1260 \catcode'\@=11 %
1261 \expandafter\ifx\csname count@\endcsname\relax
1262   \countdef\count@=255 %
1263 \fi
1264 \expandafter\ifx\csname @gobble\endcsname\relax
1265   \long\def\@gobble#1{}%
1266 \fi
1267 \expandafter\ifx\csname @firstofone\endcsname\relax
1268   \long\def\@firstofone#1{#1}%
1269 \fi
1270 \expandafter\ifx\csname loop\endcsname\relax
1271   \expandafter\@firstofone
1272 \else
1273   \expandafter\@gobble
1274 \fi
1275 {%
1276   \def\loop#1\repeat{%
1277     \def\body{#1}%
1278     \iterate
1279   }%
1280   \def\iterate{%
1281     \body
1282     \let\next\iterate
1283   \else
1284     \let\next\relax
1285   \fi
1286   \next
1287 }%
1288 \let\repeat=\fi
1289 }%
1290 \def\RestoreCatcodes{}
1291 \count@=0 %
1292 \loop
1293   \edef\RestoreCatcodes{%
1294     \RestoreCatcodes

```

```

1295 \catcode\the\count@=\the\catcode\count@\relax
1296 }%
1297 \ifnum\count@<255 %
1298 \advance\count@ 1 %
1299 \repeat
1300
1301 \def\RangeCatcodeInvalid#1#2{%
1302 \count@=#1\relax
1303 \loop
1304 \catcode\count@=15 %
1305 \ifnum\count@<#2\relax
1306 \advance\count@ 1 %
1307 \repeat
1308 }
1309 \expandafter\ifx\csname LoadCommand\endcsname\relax
1310 \def\LoadCommand{\input kvoptions.sty\relax}%
1311 \fi
1312 \def\Test{%
1313 \RangeCatcodeInvalid{0}{47}%
1314 \RangeCatcodeInvalid{58}{64}%
1315 \RangeCatcodeInvalid{91}{96}%
1316 \RangeCatcodeInvalid{123}{255}%
1317 \catcode'\@=12 %
1318 \catcode'\=0 %
1319 \catcode'\{=1 %
1320 \catcode'\}=2 %
1321 \catcode'\#=6 %
1322 \catcode'\[=12 %
1323 \catcode'\]=12 %
1324 \catcode'\%=14 %
1325 \catcode'\ =10 %
1326 \catcode13=5 %
1327 \LoadCommand
1328 \RestoreCatcodes
1329 }
1330 \Test
1331 \csname @@end\endcsname
1332 \end
1333 </test1>
1334 <*test2>
1335 \NeedsTeXFormat{LaTeX2e}
1336 \makeatletter
1337 \catcode'\@=11 %
1338 \def\RestoreCatcodes{}
1339 \count@=0 %
1340 \loop
1341 \edef\RestoreCatcodes{%
1342 \RestoreCatcodes
1343 \catcode\the\count@=\the\catcode\count@\relax
1344 }%
1345 \ifnum\count@<255 %
1346 \advance\count@\@ne
1347 \repeat
1348
1349 \def\RangeCatcodeInvalid#1#2{%
1350 \count@=#1\relax
1351 \loop
1352 \catcode\count@=15 %
1353 \ifnum\count@<#2\relax
1354 \advance\count@\@ne
1355 \repeat
1356 }

```

```

1357 \def\Test#1{%
1358   \RangeCatcodeInvalid{0}{47}%
1359   \RangeCatcodeInvalid{58}{64}%
1360   \RangeCatcodeInvalid{91}{96}%
1361   \RangeCatcodeInvalid{123}{255}%
1362   \catcode'\@=12 %
1363   \catcode'\=0 %
1364   \catcode'\{=1 %
1365   \catcode'\}=2 %
1366   \catcode'\#=6 %
1367   \catcode'\[=12 %
1368   \catcode'\]=12 %
1369   \catcode'\%=14 %
1370   \catcode'\ =10 %
1371   \catcode13=5 %
1372   #1\relax
1373   \RestoreCatcodes
1374 }
1375 \Test{\RequirePackage{kvoptions-patch}}%
1376 \Test{\RequirePackage{kvoptions}}%
1377 \csname @@end\endcsname
1378 \end{test2}

```

8 Installation

8.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/kvoptions.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/kvoptions.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 \TeX Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

8.2 Bundle installation

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

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

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

8.3 Package installation

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

```
tex kvoptions.dtx
```

¹<http://ftp.ctan.org/tex-archive/>

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

```

kvoptions.sty           → tex/latex/oberdiek/kvoptions.sty
kvoptions-patch.sty     → tex/latex/oberdiek/kvoptions-patch.sty
kvoptions.pdf           → doc/latex/oberdiek/kvoptions.pdf
example-mycolorsetup.sty → doc/latex/oberdiek/example-mycolorsetup.sty
test/kvoptions-test1.tex → doc/latex/oberdiek/test/kvoptions-test1.tex
test/kvoptions-test2.tex → doc/latex/oberdiek/test/kvoptions-test2.tex
kvoptions.dtx           → source/latex/oberdiek/kvoptions.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`.

8.4 Refresh file name databases

If your \TeX distribution (`te \TeX` , `mik \TeX` , ...) relies on file name databases, you must refresh these. For example, `te \TeX` users run `texhash` or `mktextlsr`.

8.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk kvoptions.pdf unpack_files output .
```

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{kvoptions.dtx}
```

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

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

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

An example follows how to generate the documentation with `pdf \LaTeX` :

```

pdflatex kvoptions.dtx
makeindex -s gind.ist kvoptions.idx
pdflatex kvoptions.dtx
makeindex -s gind.ist kvoptions.idx
pdflatex kvoptions.dtx

```

9 References

- [1] Package `ifthen`, David Carlisle, 2001/05/26.[CTAN:macros/latex/base/ifthen.dtx](#)
- [2] Package `helvet`, Sebastian Rahtz, Walter Schmidt, 2004/01/26.[CTAN:macros/latex/required/psnfss/psfonts.dtx](#)

- [3] Package `hyperref`, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12.[CTAN:macros/latex/contrib/hyperref/](#)
- [4] Package `keyval`, David Carlisle, 1999/03/16.[CTAN:macros/latex/required/graphics/keyval.dtx](#)
- [5] Package `multicol`, Frank Mittelbach, 2004/02/14.[CTAN:macros/latex/required/tools/multicol.dtx](#)
- [6] Package `tabularx`, David Carlisle, 1999/01/07.[CTAN:macros/latex/required/tools/tabularx.dtx](#)
- [7] Package `tracefnt`, Frank Mittelbach, Rainer Schöpf, 1997/05/29.[CTAN:macros/latex/base/ltfsstrc.dtx](#)
- [8] Package `xkeyval`, Hendri Adriaens, 2005/05/07.[CTAN:macros/latex/contrib/xkeyval/](#)
- [9] The L^AT_EX3 Project, *L^AT_EX 2_ε for class and package writers*, 2003/12/09.
[CTAN:macros/latex/doc/clsguide.pdf](#)

10 History

[0000/00/00 v0.0]

- Probably David Carlisle’s code in `hyperref` was the start.

[2004/02/22 v1.0]

- The first version was never published. It also has offered a patch to get rid of L^AT_EX’s option expansion.

[2006/02/16 v2.0]

- Now the package is redesigned with an easier user interface.
- `\ProcessKeyvalOptions` remains the central service, inherited from `hyperref`’s `\ProcessOptionsWithKV`. Now the use inside classes is also supported.
- Provides help macros for boolean and simple string options.
- Fixes for the patch of L^AT_EX. The patch is only enabled, if the user requests it.

[2006/02/20 v2.1]

- Unused option list is sanitized to prevent problems with other packages that uses own processing methods for key value options. Disadvantage: the unused global option detection is weakened.
- New option type by `\DeclareVoidOption` for options without value.
- Default rule by `\DeclareDefaultOption`.
- Dynamic options: `\DisableKeyvalOption`.

[2006/06/01 v2.2]

- Fixes for option patch.

[2006/08/17 v2.3]

- `\DeclareBooleanOption` renamed to `\DeclareBoolOption` to avoid a name clash with package `\ifoption`.

[2006/08/22 v2.4]

- Option patch: `\ExecuteOptions` does not change the meaning of macro `\CurrentOption` at all.

[2007/04/11 v2.5]

- Line ends sanitized.

[2007/05/06 v2.6]

- Uses package `etexcmds`.

[2007/06/11 v2.7]

- The patch part fixes LaTeX bug latex/3965.

[2007/10/02 v2.8]

- Compatibility for plain-TeX added.
- Typos in documentation fixed (Axel Sommerfeldt).

[2007/10/11 v2.9]

- Bug fix for option patch.

[2007/10/18 v3.0]

- New package `kvoptions-patch`.

[2009/04/10 v3.1]

- Space by line end removed in definition of internal macro.

[2009/07/17 v3.2]

- `\ProcessLocalKeyvalOptions` added.
- `\DisableKeyvalOption` with the `action=ignore` option fixed (Joseph Wright).

11 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; numbers in roman refer to the code lines where the entry is used.

Symbols	
<code>\#</code> 1259, 1321, 1366	<code>\@badrequireerror</code> 1246
<code>\%</code> 1324, 1369	<code>\@classoptionslist</code>
<code>\@</code> 1260, 1317, 1337, 1362	. . . 594, 596, 909, 927, 1013, 1015
<code>\@@fileswith@pti@ns</code> 967, 1245	<code>\@cls@pkg</code> 1074, 1109, 1124, 1127
<code>\@unprocessedoptions</code> . . . 1098, 1133	<code>\@clsextension</code> . 274, 275, 294, 314,
<code>\@SetupDriver</code> 41, 43	315, 339, 352, 385, 413, 414,
	435, 445, 466, 546, 558, 569,

539, 540, 542, 543, 632, 717, 780, 793, 835, 865, 876, 881, 884, 896, 918, 933, 964, 980, 996, 1004, 1055, 1069, 1093, 1099, 1100, 1111, 1139, 1261, 1264, 1267, 1270, 1309, 1331, 1377			
\CurrentOption . 35, 37, 38, 41, 58, 62, 64, 429, 635, 637, 639, 642, 645, 651, 656, 666, 668, 670, 673, 677, 689, 724, 726, 728, 731, 735, 747, 757, 758, 760, 769, 773, 902, 903, 913, 918, 927, 928, 930, 933, 956, 963, 964, 966, 974, 989, 999, 1003, 1004, 1009, 1056, 1141, 1142, 1147			
\CurrentOption@SaveLevel			
. 993, 996, 1000, 1001, 1006, 1007			
\CurrentOptionKey 769, 772			
\CurrentOptionValue			
. 56, 767, 768, 770, 777			
D			
\DeclareBoolOption . 5, 20, 22, <u>255</u> , 459			
\DeclareComplementaryOption			
. 5, <u>287</u> , 460			
\DeclareDefaultOption 6, 55, <u>452</u>			
\DeclareOption 223, 224			
\DeclareStringOption			
. 4, 26, 28, 50, 52, <u>365</u> , 461			
\DeclareVoidOption . . 6, 32, 33, 34, <u>401</u>			
\default@ds 959			
\define@key 230, 241,			
272, 312, 382, 411, 464, 468, 533			
\detokenize			
864, 868, 974, 976, 987, 989, 1161			
\DisableKeyvalOption 7, 472			
\do . . . 596, 637, 668, 726, 757, 902, 927, 943, 956, 963, 1003, 1141, 1226			
\documentclass 847, 1084			
\ds@ 889			
E			
\emph 48, 87, 91, 93			
\empty 109			
\end 1332			
\endcsname 105, 118, 142, 158, 165, 232, 238, 243, 249, 259, 260, 268, 306, 307, 308, 309, 328, 358, 395, 420, 483, 491, 495, 507, 511, 534, 539, 540, 542, 543, 632, 717, 780, 793, 835, 865, 877, 882, 884, 896, 918, 933, 964, 980, 996, 1004, 1055, 1069, 1093, 1099, 1100, 1111, 1139, 1261, 1264, 1267, 1270, 1309, 1331, 1377			
\endinput 127, 830			
\etex@unexpanded 602, 606, 1176, 1183, 1212, 1213, 1217, 1218			
\ExecuteOptions 994			
F			
\futurelet 1186			
G			
\gdef 647, 876, 881, 1015			
I			
\ifcase 106			
\ifetex@unexpanded 842			
\ifin@ 916, 931, 949			
\ifKV@dyn@global 489, 493, 500, 505, 509, 516, 537, 553			
\ifMCS@print 79			
\ifnum 1297, 1305, 1345, 1353			
\ifpdf 25			
\ifx 56, 88, 107, 109, 118, 142, 150, 153, 274, 294, 314, 333, 336, 339, 352, 376, 385, 413, 433, 435, 445, 526, 546, 558, 569, 592, 594, 600, 634, 641, 644, 705, 755, 758, 768, 780, 835, 903, 906, 925, 928, 944, 983, 985, 1012, 1013, 1033, 1035, 1115, 1134, 1142, 1161, 1171, 1190, 1227, 1229, 1232, 1246, 1261, 1264, 1267, 1270, 1309			
\immediate 120, 144			
\in@ 913			
\in@false 941			
\in@true 945, 950			
\input 1254, 1310			
\InputIfFileExists 1094			
\iterate 1278, 1280, 1282			
K			
\KV@@sp@def 777			
\KV@setcurrentvalue <u>776</u>			
\KV@action@error 519			
\KV@action@ignore 503			
\KV@action@undef 487			
\KV@action@warning 522			
\KV@AtEnd 178, 179, 785, 806, 807, 828, 829, 839, 849, 857, 1251			
\KV@boolkey 273, 313, <u>330</u>			
\KV@calldefault 685, 743, <u>751</u>			
\KV@checkfirstA 1195, 1198			
\KV@checkfirsttok 1186, 1189			
\KV@checkkv 1156, 1167			
\KV@CurrentOption			
. 596, 598, 607, 611, 616, 620			
\KV@DeclareStringOption 367, 369, <u>372</u>			
\KV@disable@error 557			
\KV@disable@warning 568			
\KV@do@action 520, 523, 525			
\KV@ExecuteOptions 995, 998			
\KV@false 336, <u>361</u>			
\KV@family <u>230</u> , 272, 296, 312, 382, 411, 583, 696			
\KV@fileswith@pti@ns 1011, 1245, 1248			
\KV@getkey . . . 597, 639, 670, 728, <u>750</u>			
\KV@ifdefinable 256, 257, 258, 304, 305, <u>327</u> , 373, 404			
\KV@ifempty 1155, 1158, 1160, 1168, 1199, 1202, 1210			
\KV@in@ 930, 940			
\KV@nil 1186, 1209, 1213, 1222			

\KVO@normalize ..	861, 874, 1014, 1149	\next ..	403, 405, 408, 1282, 1284, 1286
\KVO@onefilewithoptions		\numexpr	1001, 1007
.....	1019, 1024, 1030, 1037, 1051		
\KVO@param			O
..	331, 332, 333, 336, 345, 357, 358	\on@line	1107
\KVO@Patch	600, 641, 1250	\OptionNotUsed	982
\KVO@prefix	241, 256, 257, 258,		P
	260, 261, 268, 279, 288, 298,	\PackageError	
	304, 305, 306, 307, 308, 309,	289, 477, 527, 561, 707, 844
	319, 373, 374, 395, 404, 420, 424	\PackageInfo	
\KVO@process@ptions ...	922, 938, 955	..	123, 355, 388, 448, 498, 514, 549
\KVO@processOptions	899, 901	\PackageWarning ...	262, 342, 438, 572
\KVO@ProcessKeyvalOptions		\PackageWarningNoLine ..	57, 836, 852
.....	583, 587, 589	\PassOptionsToPackage	63, 80
\KVO@ProcessLocalKeyvalOptions ..		\ProcessKeyvalOptions ..	3, 74, 579, 781
.....	696, 700, 702	\ProcessLocalKeyvalOptions ..	3, 692, 708
\KVO@removeelement	1223	\ProcessOptions	229, 888
\KVO@removegarbage	1213, 1222	\protect	213
\KVO@removelastspace		\ProvidesPackage	5, 159, 832
.....	1192, 1200, 1203, 1205, 1209		
\KVO@result	1150, 1152, 1175, 1176,		R
	1182, 1183, 1211, 1212, 1216, 1217	\RangeCatcodeInvalid	
\KVO@SanitizedCurrentOption	1301, 1313, 1314, 1315,
.....	957, 980, 1123, 1126, 1146		1316, 1349, 1358, 1359, 1360, 1361
\KVO@setcurrents	760, 766	\renewcommand	93
\KVO@setcurrentvalue	773, 776	\repeat	1276, 1288, 1299, 1307, 1347, 1355
\KVO@splitcomma	1151, 1154, 1158	\RequirePackage	7, 8,
\KVO@temp	591, 601, 603,		84, 201, 226, 841, 846, 1375, 1376
	613, 614, 629, 636, 663, 667,	\reserved@a	1018, 1023,
	684, 688, 704, 714, 721, 725,		1043, 1044, 1047, 1049, 1059, 1119
	742, 746, 861, 868, 874, 878,	\reserved@b	1032, 1040, 1044
	884, 1014, 1016, 1029, 1037, 1046	\RestoreCatcodes	1290, 1293,
\KVO@true	333, 361		1294, 1328, 1338, 1341, 1342, 1373
\KVO@use@ption	917, 932, 958, 961, 970		S
\KVO@VOID@	411, 427, 433	\setkeys	44, 253, 474, 686, 744
\KVO@voidkey	412, 428	\SetupDriver	32, 33, 34, 35, 38, 40
\KVO@x	1170, 1171, 1176	\SetupKeyvalOptions ..	4, 13, 252, 455
\KVO@xprocessOptions	899, 924	\space ..	294, 1074, 1077, 1080, 1082,
\KVOdyn@action	476, 479, 483		1086, 1109, 1124, 1127, 1129, 1130
\KVOdyn@ext ...	463, 466, 470, 535, 546	\strip@prefix	
\KVOdyn@name	462, 465, 469, 526, 535, 551	..	205, 209, 479, 1068, 1073, 1147
			T
L		\t	1231, 1232
\LoadClass	1115	\Test	1312, 1330, 1357, 1375, 1376
\LoadCommand	1310, 1327	\textcolor	94
\loop	1276, 1292, 1303, 1340, 1351	\the ...	166, 167, 168, 169, 180, 382,
			391, 635, 656, 663, 666, 673,
M			677, 685, 686, 721, 724, 731,
\makeatletter	1058, 1336		735, 743, 744, 794, 795, 796,
\MCS@driver	84		797, 808, 913, 1001, 1007, 1046,
\MCS@emph	88, 94		1231, 1234, 1236, 1241, 1295, 1343
\meaning ...	205, 479, 1069, 1073, 1147	\TMP@EnsureCode ...	177, 184, 185,
\MessageBreak	58, 263, 264,		186, 187, 188, 189, 190, 191,
	290, 295, 297, 299, 300, 345,		192, 193, 194, 195, 196, 197,
	441, 479, 499, 515, 528, 552,		198, 199, 805, 812, 813, 814,
	564, 575, 845, 853, 854, 1076,		815, 816, 817, 818, 819, 820,
	1077, 1079, 1080, 1081, 1083,		821, 822, 823, 824, 825, 826, 827
	1085, 1108, 1109, 1110, 1111, 1127	\toks	627, 672, 673,
			685, 712, 730, 731, 743, 911, 913
N			
\NeedsTeXFormat	4, 788, 1335		
\newcommand	40, 43, 252, 255,		
	287, 365, 401, 452, 472, 579, 692		

<code>\toks@</code>	377, 379, 382, 384, 391, 629, 631, 635, 636, 655, 656, 662, 663, 666, 667, 676, 677, 686, 714, 716, 720, 721, 724, 725, 734, 735, 744, 907, 909, 913, 1045, 1046, 1225, 1231, 1234, 1236, 1241		
<code>\tw@</code>	627, 672, 673, 685, 712, 730, 731, 743, 911, 913		
		X	
		<code>\x</code>	105, 107, 109, 119, 123, 125, 143, 148, 158, 164, 172, 271, 282, 311, 322, 381, 398, 410, 423, 431, 433, 475, 485, 488, 504, 532, 582, 585, 695, 698, 753, 755, 792, 800, 862, 871, 912, 915, 943, 944, 948, 953, 972, 979, 1065, 1067, 1077, 1083, 1226, 1227, 1229, 1236, 1240, 1243
		Y	
		<code>\y</code>	1072, 1073, 1080, 1083
		Z	
		<code>\zap@space</code>	772, 1043, 1170, 1184
		U	
<code>\unexpanded</code>	845		
		W	
<code>\write</code>	120, 144		