

The kvoptions package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2009/08/13 v3.4

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	3
1.2	Overview	3
2	Usage	3
2.1	Process options	3
2.1.1	\ProcessKeyvalOptions	3
2.1.2	\ProcessLocalKeyvalOptions	4
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	Local options	6
2.2.7	Dynamic options	7
2.2.8	\DisableKeyvalOption	7
2.3	Global vs. local options	8
2.4	Summary of internal macros	8
2.5	plain-TeX	9
3	Example	9
4	Package options	10
4.1	Package kvoptions-patch	11
4.2	Option debugshow	12
5	Limitations	12
5.1	Compatibility	12
5.1.1	Package kvoptions-patch vs. package xkvltxp	12
5.2	Limitations	12
5.2.1	Option comparisons	12
5.2.2	Option list parsing with package kvoptions-patch	13

6	Implementation	13
6.1	Preamble	13
6.2	Option declaration macros	16
6.2.1	\SetupKeyvalOptions	16
6.2.2	\DeclareBoolOption	16
6.2.3	\DeclareStringOption	19
6.2.4	\DeclareVoidOption	19
6.2.5	\DeclareDefaultOption	20
6.2.6	\DeclareLocalOptions	21
6.3	Dynamic options	21
6.3.1	\DisableKeyvalOption	21
6.4	Process options	23
6.4.1	\ProcessKeyvalOptions	23
6.4.2	\ProcessLocalKeyvalOptions	25
6.4.3	Helper macros	26
6.5	plain-TEX	27
6.6	Package kvoptions-patch	27
7	Test	35
7.1	Preface for standard catcode check	35
7.2	Catcode checks for loading	35
8	Installation	37
8.1	Download	37
8.2	Bundle installation	37
8.3	Package installation	37
8.4	Refresh file name databases	38
8.5	Some details for the interested	38
9	References	38
10	History	39
	[0000/00/00 v0.0]	39
	[2004/02/22 v1.0]	39
	[2006/02/16 v2.0]	39
	[2006/02/20 v2.1]	39
	[2006/06/01 v2.2]	40
	[2006/08/17 v2.3]	40
	[2006/08/22 v2.4]	40
	[2007/04/11 v2.5]	40
	[2007/05/06 v2.6]	40
	[2007/06/11 v2.7]	40
	[2007/10/02 v2.8]	40
	[2007/10/11 v2.9]	40
	[2007/10/18 v3.0]	40
	[2009/04/10 v3.1]	40
	[2009/07/17 v3.2]	40
	[2009/07/21 v3.3]	40
	[2009/08/13 v3.4]	40
11	Index	41

1 Introduction

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 The beginning

This package `kvoptions` 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 L^AT_EX's package and class `options`:

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

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 L^AT_EX'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

```
\ProcessLocalKeyvalOptions {⟨family⟩}
\ProcessLocalKeyvalOptions *
```

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	⟨\@currname⟩	(foobar)	\ProcessKeyvalOptions* \DeclareBoolOption \DeclareStringOption
prefix	⟨\@currname⟩@	(foobar@)	\DeclareBoolOption \DeclareStringOption \DeclareVoidOption

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 \LaTeX 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 `\CurrentValue` 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 Local options

```
\DeclareLocalOption{<option>}
\DeclareLocalOptions{<option list>}
```

Both macros mark package options as local options. That means that they are ignored by `\ProcessKeyvalOptions` if they are given as global options. `\DeclareLocalOptions` takes one option, `\DeclareLocalOptions` expects a comma separated list of options.

2.2.7 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.

2.2.8 `\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 Global vs. local options

Options that are given for `\documentclass` are called global options. They are known to the class and all packages. A package may make use of a global option and marks it as used. The advantage for the user is the freedom to specify options both in the `\documentclass` or `\usepackage` commands.

However global options are shared with the class options and options of all other packages. Thus there can be the same option with different semantics for different packages and classes. As example, package `bookmark` knows option `open` that specifies whether the bookmarks are opened or closed initially. It's values are `true` or `false`. Since KOMA-Script version 3.00 the KOMA classes also introduces option `open` with values `right` and `any` and a complete different meaning.

Such conflicts can be resolved by marking all or part of options as local by `\DeclareLocalOption` or `\DeclareLocalOptions`. Then the packages ignores global occurrences of these options. Package `kvoptions` provides two methods:

- `\ProcessLocalKeyvalOptions` automatically uses all options as local options. It ignores all global options.
- `\DeclareLocalOption` or `\DeclareLocalOptions` marks options as local options. `\ProcessKeyvalOptions` will then ignore global occurrences for these local options.

Since version 1.5 package `bookmark` uses the latter method. It checks global and local option places for driver options and limits all other options as local options. Thus the class option `open` of KOMA-Script is not misread as option for package `bookmark`.

2.4 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 `\<prefix>` that can be configured by `\SetupKeyvalOptions`.

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

2.5 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 <*example>
2   % Package identification
3   % -----
4   \NeedsTeXFormat{LaTeX2e}
5   \ProvidesPackage{example-mycolorsetup}[2009/08/13 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 L^AT_EX 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`

L^AT_EX'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. L^AT_EX 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, L^AT_EX 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[pdauthor=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[pdauthor=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 -TeX, 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 -TeX.
- 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 `LATEX`.

```
99 \begingroup
100 \catcode44 12 % ,
101 \catcode45 12 % -
102 \catcode46 12 % .
103 \catcode58 12 % :
104 \catcode64 11 % @
105 \catcode123 1 % {
106 \catcode125 2 % }
107 \expandafter\let\expandafter\x\csname ver@kvoptions.sty\endcsname
108 \ifx\x\relax % plain-TeX, first loading
109 \else
110   \def\empty{}%
111   \ifx\x\empty % LaTeX, first loading,
112     % variable is initialized, but \ProvidesPackage not yet seen
113   \else
114     \catcode35 6 % #
115     \expandafter\ifx\csname PackageInfo\endcsname\relax
116       \def\x#1#2{%
117         \immediate\write-1{Package #1 Info: #2.}%
118       }%
119     \else
120       \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
```

```

121     \fi
122     \x{kvoptions}{The package is already loaded}%
123     \aftergroup\endinput
124   \fi
125   \fi
126 \endgroup
Package identification:
127 \begingroup
128   \catcode35 6 % #
129   \catcode40 12 % (
130   \catcode41 12 % )
131   \catcode44 12 % ,
132   \catcode45 12 % -
133   \catcode46 12 % .
134   \catcode47 12 % /
135   \catcode58 12 % :
136   \catcode64 11 % @
137   \catcode91 12 % [
138   \catcode93 12 % ]
139   \catcode123 1 % {
140   \catcode125 2 % }
141   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
142     \def\x#1#2#3[#4]{\endgroup
143       \immediate\write-1{Package: #3 #4}%
144       \xdef#1{#4}%
145     }%
146   \else
147     \def\x#1#2[#3]{\endgroup
148       #2[#{#3}]%
149       \ifx#1\@undefined
150         \xdef#1{#3}%
151       \fi
152       \ifx#1\relax
153         \xdef#1{#3}%
154       \fi
155     }%
156   \fi
157 \expandafter\x\csname ver@kvoptions.sty\endcsname
158 \ProvidesPackage{kvoptions}%
159 [2009/08/13 v3.4 Keyval support for LaTeX options (H0)]

```

Catcodes

```

160 \begingroup
161   \catcode123 1 % {
162   \catcode125 2 % }
163   \def\x{\endgroup
164     \expandafter\edef\csname KV0@AtEnd\endcsname{%
165       \catcode35 \the\catcode35\relax
166       \catcode64 \the\catcode64\relax
167       \catcode123 \the\catcode123\relax
168       \catcode125 \the\catcode125\relax
169     }%
170   }%
171 \x
172 \catcode35 6 % #
173 \catcode64 11 % @
174 \catcode123 1 % {
175 \catcode125 2 % }
176 \def\TMP@EnsureCode#1#2{%
177   \edef\KV0@AtEnd{%
178     \KV0@AtEnd

```

```

179 \catcode#1 \the\catcode#1\relax
180 }%
181 \catcode#1 #2\relax
182 }
183 \TMP@EnsureCode{1}{14}% ^^A (comment)
184 \TMP@EnsureCode{2}{14}% ^^A (comment)
185 \TMP@EnsureCode{33}{12}% !
186 \TMP@EnsureCode{39}{12}% '
187 \TMP@EnsureCode{40}{12}% (
188 \TMP@EnsureCode{41}{12}% )
189 \TMP@EnsureCode{42}{12}% *
190 \TMP@EnsureCode{44}{12}% ,
191 \TMP@EnsureCode{45}{12}% -
192 \TMP@EnsureCode{46}{12}% .
193 \TMP@EnsureCode{47}{12}% /
194 \TMP@EnsureCode{58}{12}% :
195 \TMP@EnsureCode{61}{12}% =
196 \TMP@EnsureCode{62}{12}% >
197 \TMP@EnsureCode{94}{7}% ^ (superscript)
198 \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`.

```

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

```

Macro `\DeclareLocalOptions` parses a comma separated key list and uses `\comma@parse` of package `kvsetkeys`, version 1.3.

```

202 \RequirePackage{kvsetkeys}[2007/09/29]

```

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}[2009/08/13]%
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.\@current\endcsname{#1}%
233 }
234 \def\KV0@family{%
235 \@ifundefined{KV0@family@%
236 \currname.\@current}{%
237 }{%
238 \csname KV0@family@%
239 \currname.\@current\endcsname
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.\@current\endcsname{#1}%
244 }
245 \def\KV0@prefix{%
246 \@ifundefined{KV0@prefix@%
247 \currname \@%
248 }{%
249 \csname KV0@prefix@%
250 \currname \@current\endcsname
251 }

```

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

```

252 \newcommand*{\SetupKeyvalOptions}{%
253 \setkeys{KV0}%
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   \KV0@ifdefinable{if\KV0@prefix#2}{%
257     \KV0@ifdefinable{\KV0@prefix#2true}{%
258       \KV0@ifdefinable{\KV0@prefix#2false}{%
259         \csname newif\expandafter\endcsname
260         \csname if\KV0@prefix#2\endcsname
261         \@ifundefined{\KV0@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\KV0@prefix#2#1\endcsname
269         }%
270         \begingroup
271         \edef\x{\endgroup
272           \noexpand\define@key{\KV0@family}{#2}[true]{%
273             \noexpand\KV0@boolkey{\@currname}%
274             \ifx\@currentx\@clsextension
275               \noexpand\@clsextension
276             \else
277               \noexpand\@pkgextension
278             \fi
279             {\KV0@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\KV0@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.\@currentx’.\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\@currentx\@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 accidentally 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{\KVO@family}{#1}{\KVO@VOID@}{%
412       \noexpand\KVO@voidkey{\@currname}%
413       \ifx\@current\@clsextension
414         \noexpand\@clsextension
415       \else
416         \noexpand\@pkgextension
417       \fi
418       {#1}%
419       {####1}%
420       \expandafter\noexpand\csname\KVO@prefix#1\endcsname
421     }%
422   }%
423   \x
424   \@namedef{\KVO@prefix#1}%
425 }%
426 }
427 \def\KVO@VOID@{@VOID@}

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

428 \def\KVO@voidkey#1#2#3#4{%
429   \def\CurrentOption{#3}%
430   \begingroup
431     \def\x{#4}%
432   \expandafter\endgroup
433   \ifx\x\KVO@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{\KVO@default@\@currname.\@current}%
454 }

```

6.2.6 \DeclareLocalOptions

\DeclareLocalOptions

```
455 \newcommand*{\DeclareLocalOptions}[1]{%
456   \comma@parse{#1}\KV0@DeclareLocalOption
457 }
```

\KV0@DeclareLocalOption

```
458 \def\KV0@DeclareLocalOption#1{%
459   \expandafter\def\csname KV0@local@\KV0@family @#1\endcsname{%
460 }
```

6.3 Dynamic options

6.3.1 \DisableKeyvalOption

```
461 \SetupKeyvalOptions{%
462   family=KV0dyn,%
463   prefix=KV0dyn%
464 }
465 \DeclareBoolOption[true]{global}
466 \DeclareComplementaryOption{local}{global}
467 \DeclareStringOption[undef]{action}
468 \let\KV0dyn@name\relax
469 \let\KV0dyn@ext\@empty
470 \define@key{KV0dyn}{class}{%
471   \def\KV0dyn@name{#1}%
472   \let\KV0dyn@ext\@clsextension
473 }
474 \define@key{KV0dyn}{package}{%
475   \def\KV0dyn@name{#1}%
476   \let\KV0dyn@ext\@pkgextension
477 }
478 \newcommand*{\DisableKeyvalOption}[3][ ]{%
479   \begingroup
480     \setkeys{KV0dyn}{#1}%
481     \def\x{\endgroup}%
482     \@ifundefined{KV0@action@\KV0dyn@action}{%
483       \PackageError{kvoptions}{%
484         Unknown disable action %
485         '\expandafter\strip@prefix\meaning\KV0dyn@action'\MessageBreak
486         for option '#3' in keyval family '#2'%
487       }\@ehc
488     }{%
489       \csname KV0@action@\KV0dyn@action\endcsname{#2}{#3}%
490     }%
491   \x
492 }
493 \def\KV0@action@undef#1#2{%
494   \edef\x{\endgroup
495     \ifKV0dyn@global\global\fi
496     \let
497     \expandafter\noexpand\csname KV@#1@#2\endcsname
498     \relax
499     \ifKV0dyn@global\global\fi
500     \let
501     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
502     \relax
503   }%
504   ^^A\PackageInfo{kvoptions}{%
505     ^^A [option] key '#2' of family '#1'\MessageBreak
506     ^^A is disabled (undef, \ifKV0dyn@global\global\else local\fi)%
507   ^^A}%
```

```

508 }
509 \def\KV0@action@ignore#1#2{%
510   \edef\x{\endgroup
511     \ifKV0dyn@global\global\fi
512     \let
513     \expandafter\noexpand\csname KV@#1@#2\endcsname
514     \noexpand@gobble
515     \ifKV0dyn@global\global\fi
516     \let
517     \expandafter\noexpand\csname KV@#1@#2@default\endcsname
518     \noexpand@empty
519   }%
520   ^^A\PackageInfo{kvoptions}{%
521     ^^A [option] key ‘#2’ of family ‘#1’\MessageBreak
522     ^^A is disabled (ignore, \ifKV0dyn@global\global\else local\fi)%
523   ^^A}%
524 }
525 \def\KV0@action@error{%
526   \KV0@do@action{error}%
527 }
528 \def\KV0@action@warning{%
529   \KV0@do@action{warning}%
530 }

#1 error or warning
#2 <family>
#3 <key>
531 \def\KV0@do@action#1#2#3{%
532   \ifx\KV0dyn@name\relax
533     \PackageError{kvoptions}{%
534       Action type ‘#1’ needs package/class name\MessageBreak
535       for key ‘#3’ in family ‘#2’%
536     }\@ehc
537   \else
538     \edef\x{\endgroup
539       \noexpand\define@key{#2}{#3}[]{%
540         \expandafter\noexpand\csname KV0@disable@#1\endcsname
541         {\KV0dyn@name}\noexpand\KV0dyn@ext{#3}%
542       }%
543     \ifKV0dyn@global
544       \global\let
545       \expandafter\noexpand\csname KV@#2@#3\endcsname
546       \expandafter\noexpand\csname KV@#2@#3\endcsname
547       \global\let
548       \expandafter\noexpand\csname KV@#2@#3@default\endcsname
549       \expandafter\noexpand\csname KV@#2@#3@default\endcsname
550     \fi
551   }%
552   ^^A\ifx\KV0dyn@ext\@clsextension
553     ^^A \expandafter\ClassInfo
554   ^^A\else
555     ^^A \expandafter\PackageInfo
556   ^^A\fi
557   ^^A{\KV0dyn@name}{%
558     ^^A [option] key ‘#3’ of family ‘#2’\MessageBreak
559     ^^A is disabled (#1, \ifKV0dyn@global\global\else local\fi)%
560   ^^A}%
561   \fi
562 }
563 \def\KV0@disable@error#1#2#3{%
564   \ifx#2\@clsextension
565     \expandafter\ClassError
566   \else

```

```

567 \expandafter\PackageError
568 \fi
569 {#1}{%
570 Option '#3' is given too late,\MessageBreak
571 now the option is ignored%
572 }\@ehc
573 }
574 \def\KV0@disable@warning#1#2#3{%
575 \ifx#2\@clsextension
576 \expandafter\ClassWarning
577 \else
578 \expandafter\PackageWarning
579 \fi
580 {#1}{%
581 Option '#3' is already consumed\MessageBreak
582 and has no effect%
583 }%
584 }

```

6.4 Process options

6.4.1 \ProcessKeyvalOptions

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

```

585 \newcommand*{\ProcessKeyvalOptions}{%
586 \@ifstar{%
587 \begingroup
588 \edef\x{\endgroup
589 \noexpand\KV0@ProcessKeyvalOptions{\KV0@family}}%
590 }%
591 \x
592 }%
593 \KV0@ProcessKeyvalOptions
594 }

595 \def\KV0@ProcessKeyvalOptions#1{%
596 \let\@tempc\relax
597 \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).

```

598 \ifx\@currentx\@clsextension
599 \else
600 \ifx\@classoptionslist\relax
601 \else
602 \for\KV0@CurrentOption:=\@classoptionslist\do{%
603 \@ifundefined{KV@#1}\expandafter\KV0@getkey
604 \KV0@CurrentOption=\@nil}{%
605 }{%
606 \@ifundefined{KV0@local@#1}\expandafter\KV0@getkey
607 \KV0@CurrentOption=\@nil}{%
608 \ifx\KV0@Patch Y%
609 \edef\KV0@temp{%
610 \etex@unexpanded\expandafter{%
611 \KV0@temp
612 }%
613 ,%
614 \etex@unexpanded\expandafter{%
615 \KV0@CurrentOption
616 }%
617 ,%
618 }%
619 \@onelevel@sanitize\KV0@CurrentOption
620 \else

```

```

621         \edef\KV0@temp{%
622             \KV0@temp
623             ,%
624             \KV0@CurrentOption
625             ,%
626         }%
627     \fi
628     \@expandtwoargs\@removeelement\KV0@CurrentOption
629     \@unusedoptionlist\@unusedoptionlist
630 }{%}%
631 }%
632 }%
633 \fi
634 \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`.

```

635 \begingroup
636   \toks\tw@{%
637     \@ifundefined{opt@\@currname.\@currentx}{%
638       \toks@\expandafter{\KV0@temp}%
639     }{%
640       \toks@\expandafter\expandafter\expandafter{%
641         \csname opt@\@currname.\@currentx\endcsname
642       }%
643       \ifx\@currentx\@clsextension
644         \edef\CurrentOption{\the\toks@}%
645         \toks@\expandafter{\KV0@temp}%
646         \@for\CurrentOption:=\CurrentOption\do{%
647           \@ifundefined{%
648             KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
649           }{%

```

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

```

650       \ifx\KV0@Patch Y%
651         \@onelevel@sanitize\CurrentOption
652       \fi
653       \ifx\@unusedoptionlist\@empty
654         \global\let\@unusedoptionlist\CurrentOption
655       \else
656         \expandafter\expandafter\expandafter\gdef
657         \expandafter\expandafter\expandafter\@unusedoptionlist
658         \expandafter\expandafter\expandafter{%
659           \expandafter\@unusedoptionlist
660           \expandafter,\CurrentOption
661         }%
662       \fi
663     }{%
664       \toks@\expandafter{%
665         \the\expandafter\toks@\expandafter,\CurrentOption
666       }%
667     }%
668   }%
669 \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.

```

670     \@ifundefined{KV0@default@\@currname.\@currentx}{%
671       \toks@\expandafter\expandafter\expandafter{%
672         \expandafter\KV0@temp\the\toks@
673       }%
674     }{%

```



```

675         \edef\CurrentOption{\the\toks@}%
676         \toks@\expandafter{\KV0@temp}%
677         \@for\CurrentOption:=\CurrentOption\do{%
678             \ifundefined{%
679                 KV@#1@\expandafter\KV0@getkey\CurrentOption=\@nil
680             }{%
681                 \toks\tw@\expandafter{%
682                     \the\toks\expandafter\tw@\expandafter,\CurrentOption
683                 }%
684             }{%
685                 \toks@\expandafter{%
686                     \the\expandafter\toks@\expandafter,\CurrentOption
687                 }%
688             }%
689         }%
690     }%
691     \fi
692 }%
693 \edef\KV0@temp{\endgroup
694     \noexpand\KV0@calldefault{\the\toks\tw@}%
695     \noexpand\setkeys{#1}{\the\toks@}%
696 }%
697 \KV0@temp

```

Some cleanup of \ProcessOptions.

```

698 \let\CurrentOption\@empty
699 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
700 }

```

6.4.2 \ProcessLocalKeyvalOptions

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

```

701 \newcommand*{\ProcessLocalKeyvalOptions}{%
702     \ifstar{%
703         \begingroup
704         \edef\x{\endgroup
705             \noexpand\KV0@ProcessLocalKeyvalOptions{\KV0@family}%
706         }%
707         \x
708     }%
709     \KV0@ProcessLocalKeyvalOptions
710 }

711 \def\KV0@ProcessLocalKeyvalOptions#1{%
712     \let\@tempc\relax
713     \let\KV0@temp\@empty

```

Check if \ProcessLocalKeyvalOptions is called inside a package.

```

714     \ifx\@current\@pkgextension
715     \else
716         \PackageError{kvoptions}{%
717             \string\ProcessLocalKeyvalOptions is intended for packages only%
718         }\@ehc
719     \fi

```

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

```

720     \begingroup
721     \toks\tw@{%
722         \@ifundefined{opt@\@currname.\@current}{%
723             \toks@\expandafter{\KV0@temp}%
724         }{%
725             \toks@\expandafter\expandafter\expandafter{%
726                 \csname opt@\@currname.\@current\endcsname
727             }%

```

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.

```

728     \@ifundefined{KV0@default@\@currname.\@currentx}{%
729         \toks@\expandafter\expandafter\expandafter{%
730             \expandafter\KV0@temp\the\toks@
731         }%
732     }{%
733         \edef\CurrentOption{\the\toks@}%
734         \toks@\expandafter{\KV0@temp}%
735         \@for\CurrentOption:=\CurrentOption\do{%
736             \@ifundefined{%
737                 KV@#1\expandafter\KV0@getkey\CurrentOption=\@nil
738             }{%
739                 \toks\tw@\expandafter{%
740                     \the\toks\expandafter\tw@\expandafter,\CurrentOption
741                 }%
742             }{%
743                 \toks@\expandafter{%
744                     \the\expandafter\toks@\expandafter,\CurrentOption
745                 }%
746             }%
747         }%
748     }%
749     \fi
750 }%
751 \edef\KV0@temp{\endgroup
752     \noexpand\KV0@calldefault{\the\toks\tw@}%
753     \noexpand\setkeys{#1}{\the\toks@}%
754 }%
755 \KV0@temp

```

Some cleanup of `\ProcessOptions`.

```

756 \let\CurrentOption\@empty
757 \AtEndOfPackage{\let\@unprocessedoptions\relax}%
758 }

```

6.4.3 Helper macros

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

```

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

```

`\KV0@calldefault`

```

760 \def\KV0@calldefault#1{%
761     \begingroup
762     \def\x{#1}%
763     \expandafter\endgroup
764     \ifx\x\@empty
765     \else
766         \@for\CurrentOption:=#1\do{%
767             \ifx\CurrentOption\@empty
768             \else
769                 \expandafter\KV0@setcurrents\CurrentOption=\@nil
770                 \@nameuse{KV0@default@\@currname.\@currentx}%
771             \fi
772         }%
773     \fi
774 }

```

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

```

775 \def\KV0@setcurrents#1=#2\@nil{%
776     \def\CurrentOptionValue{#2}%

```

```

777 \ifx\CurrentOptionValue\@empty
778   \let\CurrentOptionKey\CurrentOption
779   \let\CurrentOptionValue\relax
780 \else
781   \edef\CurrentOptionKey{\zap@space#1 \@empty}%
782   \expandafter\KV@setcurrentvalue\CurrentOption\@nil
783 \fi
784 }

```

`\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.

```

785 \def\KV@setcurrentvalue#1=#2\@nil{%
786   \KV@@sp@def\CurrentOptionValue{#2}%
787 }

```

6.5 plain-TeX

Disable L^AT_EX stuff.

```

788 \begingroup\expandafter\expandafter\expandafter\endgroup
789 \expandafter\ifx\csname documentclass\endcsname\relax
790   \def\ProcessKeyvalOptions{%
791     \@ifstar{}\@gobble
792   }%
793 \fi

794 \KV@AtEnd
795 </package>

```

6.6 Package kvoptions-patch

```

796 <*patch>
797 \NeedsTeXFormat{LaTeX2e}
798 \begingroup
799   \catcode123 1 % {
800   \catcode125 2 % }
801   \def\x{\endgroup
802     \expandafter\edef\csname KV@AtEnd\endcsname{%
803       \catcode35 \the\catcode35\relax
804       \catcode64 \the\catcode64\relax
805       \catcode123 \the\catcode123\relax
806       \catcode125 \the\catcode125\relax
807     }%
808   }%
809   \x
810   \catcode35 6 % #
811   \catcode64 11 % @
812   \catcode123 1 % {
813   \catcode125 2 % }
814   \def\TMP@EnsureCode#1#2{%
815     \edef\KV@AtEnd{%
816       \KV@AtEnd
817       \catcode#1 \the\catcode#1\relax
818     }%
819     \catcode#1 #2\relax
820   }
821   \TMP@EnsureCode{39}{12}% '
822   \TMP@EnsureCode{40}{12}% (
823   \TMP@EnsureCode{41}{12}% )
824   \TMP@EnsureCode{43}{12}% +
825   \TMP@EnsureCode{44}{12}% ,
826   \TMP@EnsureCode{45}{12}% -
827   \TMP@EnsureCode{46}{12}% .

```

```

828 \TMP@EnsureCode{47}{12}% /
829 \TMP@EnsureCode{58}{12}% :
830 \TMP@EnsureCode{60}{12}% <
831 \TMP@EnsureCode{61}{12}% =
832 \TMP@EnsureCode{62}{12}% >
833 \TMP@EnsureCode{91}{12}% [
834 \TMP@EnsureCode{93}{12}% ]
835 \TMP@EnsureCode{96}{12}% ‘
836 \TMP@EnsureCode{124}{12}% |
837 \edef\KVO@AtEnd{%
838   \KVO@AtEnd
839   \noexpand\endinput
840 }
841 \ProvidesPackage{kvoptions-patch}%
842 [2009/08/13 v3.4 LaTeX patch for keyval options (HO)]%

Check for  $\varepsilon$ -TeX.
843 \begingroup\expandafter\expandafter\expandafter\endgroup
844 \expandafter\ifx\csname eTeXversion\endcsname\relax
845   \PackageWarningNoLine{kvoptions-patch}{%
846     Package loading is aborted, because e-TeX is missing%
847   }%
848   \expandafter\KVO@AtEnd
849 \fi

Package etexcmds for \etex@unexpanded.
850 \RequirePackage{etexcmds}[2007/09/09]
851 \ifetex@unexpanded
852 \else
853   \PackageError{kvoptions-patch}{%
854     Could not find eTeX's \string\unexpanded.\MessageBreak
855     Try adding \string\RequirePackage\string{etexcmds\string} %
856     before \string\documentclass%
857   }\@ehd
858   \expandafter\KVO@AtEnd
859 \fi

Check for package xkvltxp.
860 \@ifpackageloaded{xkvltxp}{%
861   \PackageWarningNoLine{kvoptions}{%
862     Option ‘patch’ cannot be used together with\MessageBreak
863     package ‘xkvltxp’ that is already loaded.\MessageBreak
864     Therefore package loading is aborted%
865   }%
866   \KVO@AtEnd
867 }{}

868 \def\@if@options#1#2#3{%
869   \begingroup
870     \KVO@normalize\KVO@temp{#3}%
871     \edef\x{\endgroup
872       \noexpand\@if@ptions{%
873         \detokenize\expandafter\expandafter\expandafter{%
874           \csname opt@#2.#1\endcsname
875         }%
876       }{%
877         \detokenize\expandafter{\KVO@temp}%
878       }%
879     }%
880   \x
881 }

882 \def\@pass@options#1#2#3{%
883   \KVO@normalize\KVO@temp{#2}%
884   \@ifundefined{opt@#3.#1}{%
885     \expandafter\gdef\csname opt@#3.#1%

```

```

886         \expandafter\endcsname\expandafter{%
887         \KV0@temp
888     }%
889 }{%
890     \expandafter\gdef\csname opt@#3.#1%
891         \expandafter\expandafter\expandafter\endcsname
892         \expandafter\expandafter\expandafter{%
893         \csname opt@#3.#1\expandafter\endcsname\expandafter,\KV0@temp
894     }%
895 }%
896 }
897 \def\ProcessOptions{%
898     \let\ds@\empty
899     \ifundefined{opt@\currname.\@currentx}{%
900         \let\@curroptions\empty
901     }{%
902         \expandafter\expandafter\expandafter\def
903         \expandafter\expandafter\expandafter\@curroptions
904         \expandafter\expandafter\expandafter{%
905         \csname opt@\currname.\@currentx\endcsname
906     }%
907 }%
908 \ifstar\KV0@xprocess@options\KV0@process@options
909 }
910 \def\KV0@process@options{%
911     \for\CurrentOption:=\@declaredoptions\do{%
912         \ifx\CurrentOption\empty
913         \else
914             \begingroup
915                 \ifx\@currentx\@clsextension
916                     \toks@{}%
917                 \else
918                     \toks@\expandafter{\@classoptionslist,%}
919                 \fi
920                 \toks\tw@\expandafter{\@curroptions}%
921                 \edef\x{\endgroup
922                     \noexpand\in@{\CurrentOption,}{,\the\toks@\the\toks\tw@,%}%
923                 }%
924                 \x
925                 \ifin@
926                     \KV0@use@option
927                     \expandafter\let\csname ds@\CurrentOption\endcsname\empty
928                 \fi
929             \fi
930         }%
931     \KV0@process@ptions
932 }
933 \def\KV0@xprocess@options{%
934     \ifx\@currentx\@clsextension
935     \else
936         \for\CurrentOption:=\@classoptionslist\do{%
937             \ifx\CurrentOption\empty
938             \else
939                 \KV0@in@\CurrentOption\@declaredoptions
940                 \ifin@
941                     \KV0@use@option
942                     \expandafter\let\csname ds@\CurrentOption\endcsname\empty
943                 \fi
944             \fi
945         }%
946     \fi
947     \KV0@process@ptions

```

```

948 }
949 \def\KVO@in@#1#2{%
950   \in@false
951   \begingroup
952     \for\x:=#2\do{%
953       \ifx\x#1\relax
954         \in@true
955       \fi
956     }%
957   \edef\x{\endgroup
958     \ifin@
959       \noexpand\in@true
960     \fi
961   }%
962   \x
963 }

964 \def\KVO@process@ptions{%
965   \for\CurrentOption:=\@curroptions\do{%
966     \ifundefined{ds@\KVO@SanitizedCurrentOption}{%
967       \KVO@use@option
968       \default@ds
969     }%
970     \KVO@use@option
971   }%
972   \for\CurrentOption:=\@declaredoptions\do{%
973     \expandafter\let\csname ds@\CurrentOption\endcsname\relax
974   }%
975   \let\CurrentOption@empty
976   \let\@fileswithptions\@fileswithptions
977   \AtEndOfPackage{\let\@unprocessedoptions\relax}%
978 }

979 \def\KVO@use@option{%
980   \begingroup
981     \edef\x{\endgroup
982       \noexpand\@removeelement{%
983         \detokenize\expandafter{\CurrentOption}%
984       }{%
985         \detokenize\expandafter{\@unusedoptionlist}%
986       }%
987     }%
988     \x\@unusedoptionlist
989     \csname ds@\KVO@SanitizedCurrentOption\endcsname
990 }

991 \def\OptionNotUsed{%
992   \ifx\@current\@clsextension
993     \xdef\@unusedoptionlist{%
994       \ifx\@unusedoptionlist\@empty
995         \else
996           \detokenize\expandafter{\@unusedoptionlist},}%
997     \fi
998     \detokenize\expandafter{\CurrentOption}%
999   }%
1000   \fi
1001 }

Variant of \ExecuteOptions that better protects \CurrentOption.
1002 \def\CurrentOption@SaveLevel{0}
1003 \def\ExecuteOptions{%
1004   \expandafter\KVO@ExecuteOptions
1005     \csname CurrentOption@\CurrentOption@SaveLevel\endcsname
1006 }
1007 \def\KVO@ExecuteOptions#1#2{%

```

```

1008 \let#1\CurrentOption
1009 \edef\CurrentOption@SaveLevel{%
1010   \the\numexpr\CurrentOption@SaveLevel+1%
1011 }%
1012 \@for\CurrentOption:=#2\do{%
1013   \csname ds@\CurrentOption\endcsname
1014 }%
1015 \edef\CurrentOption@SaveLevel{%
1016   \the\numexpr\CurrentOption@SaveLevel-1%
1017 }%
1018 \let\CurrentOption#1%
1019 }

1020 \def\KVO@fileswithoptions#1[#2]#3[#4]{%
1021   \ifx#1\clsextension
1022     \ifx\classoptionslist\relax
1023       \KVO@normalize\KVO@temp{#2}%
1024       \expandafter\gdef\expandafter\classoptionslist\expandafter{%
1025         \KVO@temp
1026       }%
1027       \def\reserved@a{%
1028         \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
1029         \@documentclasshook
1030       }%
1031     \else
1032       \def\reserved@a{%
1033         \KVO@onefilewithoptions{#3}[#{#2}][#{#4}]#1%
1034       }%
1035     \fi
1036   \else
1037     \begingroup
1038     \let\KVO@temp\relax
1039     \let\KVO@onefilewithoptions\relax
1040     \let\@pkgextension\relax
1041     \def\reserved@b##1,{%
1042       \ifx\@nil##1\relax
1043       \else
1044         \ifx\relax##1\relax
1045         \else
1046           \KVO@onefilewithoptions{##1}[{\KVO@temp}][#{#4}]%
1047           \@pkgextension
1048         \fi
1049         \expandafter\reserved@b
1050       \fi
1051     }%
1052     \edef\reserved@a{\zap@space#3 \@empty}%
1053     \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
1054     \toks@{#2}%
1055     \def\KVO@temp{\the\toks@}%
1056     \edef\reserved@a{\endgroup \reserved@a}%
1057   \fi
1058   \reserved@a
1059 }

1060 \def\KVO@onefilewithoptions#1[#2]#3[#4]{%
1061   \@pushfilename
1062   \xdef\@currname{#1}%
1063   \global\let\@currentx#4%
1064   \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname\@empty
1065   \let\CurrentOption\@empty
1066   \@resetoptions
1067   \makeatletter
1068   \def\reserved@a{%
1069     \ifl@aded\@currentx{#1}{%

```

```

1070 \if@options\@currentx{#1}{#2}{%
1071 }{%
1072 \begingroup
1073 \ifundefined{opt@#1.\@currentx}{%
1074 \def\x{%
1075 }{%
1076 \edef\x{%
1077 \expandafter\expandafter\expandafter\strip@prefix
1078 \expandafter\meaning\csname opt@#1.\@currentx\endcsname
1079 }%
1080 }%
1081 \def\y{#2}%
1082 \edef\y{\expandafter\strip@prefix\meaning\y}%
1083 \@latex@error{Option clash for \cls@pkg\space #1}{%
1084 The package #1 has already been loaded %
1085 with options:\MessageBreak
1086 \space\space[\x]\MessageBreak
1087 There has now been an attempt to load it %
1088 with options:\MessageBreak
1089 \space\space[\y]\MessageBreak
1090 Adding the global options:\MessageBreak
1091 \space\space
1092 \x,\y\MessageBreak
1093 to your \noexpand\documentclass declaration may fix this.%
1094 \MessageBreak
1095 Try typing \space <return> \space to proceed.%
1096 }%
1097 \endgroup
1098 }%
1099 }{%
1100 \@pass@options\@currentx{#2}{#1}%
1101 \global\expandafter
1102 \let\csname ver@\@currname.\@currentx\endcsname\@empty
1103 \InputIfFileExists
1104 {\@currname.\@currentx}%
1105 {}%
1106 {\@missingfileerror\@currname\@currentx}%
1107 \let\@unprocessedoptions\@unprocessedoptions
1108 \csname\@currname.\@currentx-h@@k\endcsname
1109 \expandafter\let\csname\@currname.\@currentx-h@@k\endcsname
1110 \undefined
1111 \@unprocessedoptions
1112 }%
1113 \@ifl@ter\@currentx{#1}{#3}{%
1114 }{%
1115 \@latex@warning@no@line{%
1116 You have requested,\on@line, %
1117 version\MessageBreak
1118 #3' of \cls@pkg\space #1,\MessageBreak
1119 but only version\MessageBreak
1120 '\csname ver@#1.\@currentx\endcsname'\MessageBreak
1121 is available%
1122 }%
1123 }%
1124 \ifx\@currentx\cls@extension\let\LoadClass\@twoloadclasserror\fi
1125 \popfilename
1126 \@reset@options
1127 }%
1128 \reserved@a
1129 }

1130 \def\@unknownoptionerror{%
1131 \@latex@error{%

```



```

1132     Unknown option '\KV0@SanitizedCurrentOption' %
1133     for \@cls@pkg\space'\@currname'%
1134 }{%
1135     The option '\KV0@SanitizedCurrentOption' was not declared in %
1136     \@cls@pkg\space'\@currname', perhaps you\MessageBreak
1137     misspelled its name. %
1138     Try typing \space <return> %
1139     \space to proceed.%
1140 }%
1141 }

1142 \def\@@unprocessedoptions{%
1143     \ifx\@current\@pkgextension
1144         \ifundefined{opt@\@currname.\@current}{%
1145             \let\@curroptions\@empty
1146         }{%
1147             \expandafter\let\expandafter\@curroptions
1148                 \csname opt@\@currname.\@current\endcsname
1149         }%
1150         \@for\CurrentOption:=\@curroptions\do{%
1151             \ifx\CurrentOption\@empty\else\@unknownoptionerror\fi
1152         }%
1153     \fi
1154 }

1155 \def\KV0@SanitizedCurrentOption{%
1156     \expandafter\strip@prefix\meaning\CurrentOption
1157 }

    Normalize option list.
1158 \def\KV0@normalize#1#2{%
1159     \let\KV0@result\@empty
1160     \KV0@splitcomma#2,\@nil
1161     \let#1\KV0@result
1162 }
1163 \def\KV0@splitcomma#1,#2\@nil{%
1164     \KV0@ifempty{#1}{}%
1165     \KV0@checkkv#1=\@nil
1166 }%
1167 \KV0@ifempty{#2}{}\KV0@splitcomma#2\@nil}%
1168 }
1169 \def\KV0@ifempty#1{%
1170     \expandafter\ifx\expandafter\\\detokenize{#1}\\\%
1171         \expandafter\@firstoftwo
1172     \else
1173         \expandafter\@secondoftwo
1174     \fi
1175 }
1176 \def\KV0@checkkv#1=#2\@nil{%
1177     \KV0@ifempty{#2}{%
1178         % option without value
1179         \edef\KV0@x{\zap@space#1 \@empty}%
1180         \ifx\KV0@x\@empty
1181             % ignore empty option
1182         \else
1183             % append to list
1184             \edef\KV0@result{%
1185                 \etex@unexpanded\expandafter{\KV0@result},\KV0@x
1186             }%
1187         \fi
1188     }{%
1189         % #1: "key", #2: "value="
1190         % add key part
1191         \edef\KV0@result{%

```

```

1192     \etex@unexpanded\expandafter{\KV0@result},%
1193     \zap@space#1 \@empty
1194 }%
1195 \futurelet\@let@token\KV0@checkfirsttok#2 \@nil| = \@nil|\KV0@nil
1196 }%
1197 }
1198 \def\KV0@checkfirsttok{%
1199   \ifx\@let@token\bgroup
1200     % no space at start
1201     \expandafter\KV0@removelastspace\expandafter=%
1202     % "<value><spaceopt>= \@nil"
1203   \else
1204     \expandafter\KV0@checkfirstA
1205   \fi
1206 }
1207 \def\KV0@checkfirstA#1 #2\@nil{%
1208   \KV0@ifempty{#2}{%
1209     \KV0@removelastspace=#1 \@nil
1210   }{%
1211     \KV0@ifempty{#1}{%
1212       \KV0@removelastspace=#2\@nil
1213     }{%
1214       \KV0@removelastspace=#1 #2\@nil
1215     }%
1216   }%
1217 }
1218 \def\KV0@removelastspace#1 = \@nil|#2\KV0@nil{%
1219   \KV0@ifempty{#2}{%
1220     \edef\KV0@result{%
1221       \etex@unexpanded\expandafter{\KV0@result}%
1222       \etex@unexpanded\expandafter{\KV0@removegarbage#1\KV0@nil}%
1223     }%
1224   }{%
1225     \edef\KV0@result{%
1226       \etex@unexpanded\expandafter{\KV0@result}%
1227       \etex@unexpanded{#1}%
1228     }%
1229   }%
1230 }
1231 \def\KV0@removegarbage#1= \@nil#2\KV0@nil{#1}%

```

Arguments #1 and #2 are macros.

```

1232 \def\KV0@removeelement#1#2{%
1233   \begingroup
1234     \toks@={}%
1235     \@for\x:=#2\do{%
1236       \ifx\x\@empty
1237       \else
1238         \ifx\x#1\relax
1239       \else
1240         \edef\t{\the\toks@}%
1241         \ifx\t\@empty
1242       \else
1243         \toks@\expandafter{\the\toks@,%}%
1244       \fi
1245       \toks@\expandafter{\the\expandafter\toks@\x}%
1246     \fi
1247   \fi
1248 }%
1249 \edef\x{\endgroup
1250   \def\noexpand#2{\the\toks@}%
1251 }%
1252 \x

```

```

1253 }
1254 \let\@@fileswith@pti@ns\KV0@fileswith@pti@ns
1255 \ifx\@fileswith@pti@ns\@badrequireerror
1256 \else
1257   \let\@fileswith@pti@ns\KV0@fileswith@pti@ns
1258 \fi

\KV0@Patch

1259 \let\KV0@Patch=Y

1260 \KV0@AtEnd
1261 </patch>

```

7 Test

7.1 Preface for standard catcode check

```

1262 <*test1>
1263 \input miniltx.tex\relax
1264 </test1>

```

7.2 Catcode checks for loading

```

1265 <*test1>

1266 \catcode'\{=1 %
1267 \catcode'\}=2 %
1268 \catcode'\#=6 %
1269 \catcode'\@=11 %
1270 \expandafter\ifx\csname count@\endcsname\relax
1271   \countdef\count@=255 %
1272 \fi
1273 \expandafter\ifx\csname @gobble\endcsname\relax
1274   \long\def\@gobble#1{}%
1275 \fi
1276 \expandafter\ifx\csname @firstofone\endcsname\relax
1277   \long\def\@firstofone#1{#1}%
1278 \fi
1279 \expandafter\ifx\csname loop\endcsname\relax
1280   \expandafter\@firstofone
1281 \else
1282   \expandafter\@gobble
1283 \fi
1284 {%
1285   \def\loop#1\repeat{%
1286     \def\body{#1}%
1287     \iterate
1288   }%
1289   \def\iterate{%
1290     \body
1291     \let\next\iterate
1292   \else
1293     \let\next\relax
1294   \fi
1295   \next
1296 }%
1297 \let\repeat=\fi
1298 }%
1299 \def\RestoreCatcodes{}
1300 \count@=0 %
1301 \loop
1302   \edef\RestoreCatcodes{%
1303     \RestoreCatcodes

```

```

1304 \catcode\the\count@=\the\catcode\count@\relax
1305 }%
1306 \ifnum\count@<255 %
1307 \advance\count@ 1 %
1308 \repeat
1309
1310 \def\RangeCatcodeInvalid#1#2{%
1311 \count@=#1\relax
1312 \loop
1313 \catcode\count@=15 %
1314 \ifnum\count@<#2\relax
1315 \advance\count@ 1 %
1316 \repeat
1317 }
1318 \expandafter\ifx\csname LoadCommand\endcsname\relax
1319 \def\LoadCommand{\input kvoptions.sty\relax}%
1320 \fi
1321 \def\Test{%
1322 \RangeCatcodeInvalid{0}{47}%
1323 \RangeCatcodeInvalid{58}{64}%
1324 \RangeCatcodeInvalid{91}{96}%
1325 \RangeCatcodeInvalid{123}{255}%
1326 \catcode'\@=12 %
1327 \catcode'\=0 %
1328 \catcode'\{=1 %
1329 \catcode'\}=2 %
1330 \catcode'\#=6 %
1331 \catcode'\[=12 %
1332 \catcode'\]=12 %
1333 \catcode'\%=14 %
1334 \catcode'\ =10 %
1335 \catcode13=5 %
1336 \LoadCommand
1337 \RestoreCatcodes
1338 }
1339 \Test
1340 \csname @@end\endcsname
1341 \end
1342 </test1>
1343 <*test2>
1344 \NeedsTeXFormat{LaTeX2e}
1345 \makeatletter
1346 \catcode'\@=11 %
1347 \def\RestoreCatcodes{}
1348 \count@=0 %
1349 \loop
1350 \edef\RestoreCatcodes{%
1351 \RestoreCatcodes
1352 \catcode\the\count@=\the\catcode\count@\relax
1353 }%
1354 \ifnum\count@<255 %
1355 \advance\count@\@ne
1356 \repeat
1357
1358 \def\RangeCatcodeInvalid#1#2{%
1359 \count@=#1\relax
1360 \loop
1361 \catcode\count@=15 %
1362 \ifnum\count@<#2\relax
1363 \advance\count@\@ne
1364 \repeat
1365 }

```

```

1366 \def\Test#1{%
1367   \RangeCatcodeInvalid{0}{47}%
1368   \RangeCatcodeInvalid{58}{64}%
1369   \RangeCatcodeInvalid{91}{96}%
1370   \RangeCatcodeInvalid{123}{255}%
1371   \catcode'\@=12 %
1372   \catcode'\=0 %
1373   \catcode'\{=1 %
1374   \catcode'\}=2 %
1375   \catcode'\#=6 %
1376   \catcode'\[=12 %
1377   \catcode'\]=12 %
1378   \catcode'\%=14 %
1379   \catcode'\ =10 %
1380   \catcode13=5 %
1381   #1\relax
1382   \RestoreCatcodes
1383 }
1384 \Test{\RequirePackage{kvoptions-patch}}%
1385 \Test{\RequirePackage{kvoptions}}%
1386 \csname @@end\endcsname
1387 </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] A guide to key-value methods, Joseph Wright, second draft for `TUG-Boat`, 2009-03-17. <http://www.texdev.net/wp-content/uploads/2009/03/keyval.pdf>

- [2] Package `ifthen`, David Carlisle, 2001/05/26. [CTAN:macros/latex/base/ifthen.dtx](#)
- [3] Package `helvet`, Sebastian Rahtz, Walter Schmidt, 2004/01/26. [CTAN:macros/latex/required/psnfss/psfonts.dtx](#)
- [4] Package `hyperref`, Sebastian Rahtz, Heiko Oberdiek, 2006/02/12. [CTAN:macros/latex/contrib/hyperref/](#)
- [5] Package `keyval`, David Carlisle, 1999/03/16. [CTAN:macros/latex/required/graphics/keyval.dtx](#)
- [6] Package `multicol`, Frank Mittelbach, 2004/02/14. [CTAN:macros/latex/required/tools/multicol.dtx](#)
- [7] Package `tabularx`, David Carlisle, 1999/01/07. [CTAN:macros/latex/required/tools/tabularx.dtx](#)
- [8] Package `tracefnt`, Frank Mittelbach, Rainer Schöpf, 1997/05/29. [CTAN:macros/latex/base/ltfsstrc.dtx](#)
- [9] Package `xkeyval`, Hendri Adriaens, 2005/05/07. [CTAN:macros/latex/contrib/xkeyval/](#)
- [10] 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).

[2009/07/21 v3.3]

- `\DeclareLocalOption`, `\DeclareLocalOptions` added.

[2009/08/13 v3.4]

- Documentation addition: recommendation for Joseph Wright's review article.
- Documentation addition: local/global options.

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	
\#	1268, 1330, 1375
\%	1333, 1378
\@	1269, 1326, 1346, 1371
\@fileswith@pti@ns	976, 1254
\@unprocessedoptions	1107, 1142
\@SetupDriver	41, 43
\@badrequireerror	1255
\@classoptionslist	600, 602, 918, 936, 1022, 1024
\@cls@pkg	1083, 1118, 1133, 1136
\@clsextension	274, 275, 294, 314, 315, 339, 352, 385, 413, 414, 435, 445, 472, 552, 564, 575, 598, 643, 915, 934, 992, 1021, 1124
\@currentext	221, 232, 235, 238, 243, 246, 249, 274, 294, 295, 314, 385, 413, 453, 598, 637, 641, 643, 670, 714, 722, 726, 728, 770, 899, 905, 915, 934, 992, 1063, 1064, 1069, 1070, 1073, 1078, 1100, 1102, 1104, 1106, 1108, 1109, 1113, 1120, 1124, 1143, 1144, 1148
\@currname	57, 218, 232, 235, 236, 238, 243, 246, 247, 249, 273, 295, 313, 390, 412, 453, 637, 641, 670, 722, 726, 728, 770, 899, 905, 1062, 1064, 1102, 1104, 1106, 1108, 1109, 1133, 1136, 1144, 1148
\@curroptions	900, 903, 920, 965, 1145, 1147, 1150
\@declaredoptions	911, 939, 972
\@documentclasshook	1029
\@ehc	301, 487, 536, 572, 718
\@ehd	857
\@empty	88, 89, 469, 518, 597, 653, 698, 713, 756, 764, 767, 777, 781, 898, 900, 912, 927, 937, 942, 975, 994, 1052, 1064, 1065, 1102, 1145, 1151, 1159, 1179, 1180, 1193, 1236, 1241
\@expandtwoargs	628
\@fileswith@pti@ns	976, 1255, 1257
\@firstofone	334, 337, 405, 1277, 1280
\@firstoftwo	1171
\@for	602, 646, 677, 735, 766, 911, 936, 952, 965, 972, 1012, 1150, 1235
\@gobble	348, 514, 791, 1274, 1282
\@gobbletwo	403
\@if@pti@ns	872
\@if@ptions	868, 1070
\@ifdefinable	328
\@ifl@aded	1069
\@ifl@ter	1113
\@ifnextchar	366
\@ifpackageloaded	860
\@ifstar	586, 702, 791, 908
\@ifundefined	199, 203, 208, 211, 217, 220, 235, 246, 261, 288, 482, 603, 606, 637, 647, 670, 678, 722, 728, 736, 884, 899, 966, 1073, 1144
\@latex@error	1083, 1131
\@latex@warning@no@line	1115
\@let@token	1195, 1199
\@missingfileerror	1106
\@namedef	374, 424, 453
\@nameuse	770
\@one	223, 1355, 1363
\@nil	604, 607, 648, 679, 737, 759, 769, 775, 782, 785, 1042, 1053, 1160, 1163, 1165, 1167, 1176, 1195, 1202, 1207, 1209, 1212, 1214, 1218, 1231
\@onelevel@sanitize	204, 332, 363, 364, 619, 651
\@pass@ptions	882, 1100
\@pkgextension	277, 317, 416, 476, 714, 1040, 1047, 1143
\@popfilename	1125
\@pushfilename	1061
\@removeelement	628, 982
\@reset@ptions	1066, 1126
\@secondoftwo	1173
\@tempc	596, 712
\@twoloadclasserror	1124
\@typeset@protect	215
\@undefined	149, 1110
\@unknownoptionerror	68, 1130, 1151
\@unprocessedoptions	699, 757, 977, 1107, 1111
\@unusedoptionlist	629, 653, 654, 657, 659, 985, 988, 993, 994, 996
\@x@protect	212
\[1331, 1376
\]	376, 1170, 1327, 1372
\{	1266, 1328, 1373
\}	1267, 1329, 1374
\]	1332, 1377
_	1334, 1379
A	
\advance	1307, 1315, 1355, 1363
\aftergroup	123
\AtEndOfPackage	225, 699, 757, 977
B	
\body	1286, 1290

C		
\catcode	100,	
	101, 102, 103, 104, 105, 106,	
	114, 128, 129, 130, 131, 132,	
	133, 134, 135, 136, 137, 138,	
	139, 140, 161, 162, 165, 166,	
	167, 168, 172, 173, 174, 175,	
	179, 181, 223, 799, 800, 803,	
	804, 805, 806, 810, 811, 812,	
	813, 817, 819, 1266, 1267, 1268,	
	1269, 1304, 1313, 1326, 1327,	
	1328, 1329, 1330, 1331, 1332,	
	1333, 1334, 1335, 1346, 1352,	
	1361, 1371, 1372, 1373, 1374,	
	1375, 1376, 1377, 1378, 1379, 1380	
\ClassError	565	
\ClassInfo	353, 386, 446, 553	
\ClassWarning	340, 436, 576	
\comma@parse	456	
\count@	1271,	
	1300, 1304, 1306, 1307, 1311,	
	1313, 1314, 1315, 1348, 1352,	
	1354, 1355, 1359, 1361, 1362, 1363	
\countdef	1271	
\csname	107, 115, 141,	
	157, 164, 231, 238, 242, 249,	
	259, 260, 268, 306, 307, 308,	
	309, 328, 358, 395, 420, 459,	
	489, 497, 501, 513, 517, 540,	
	545, 546, 548, 549, 641, 726,	
	789, 802, 844, 874, 885, 890,	
	893, 905, 927, 942, 973, 989,	
	1005, 1013, 1064, 1078, 1102,	
	1108, 1109, 1120, 1148, 1270,	
	1273, 1276, 1279, 1318, 1340, 1386	
\CurrentOption	35, 37, 38, 41, 58,	
	62, 64, 429, 644, 646, 648, 651,	
	654, 660, 665, 675, 677, 679,	
	682, 686, 698, 733, 735, 737,	
	740, 744, 756, 766, 767, 769,	
	778, 782, 911, 912, 922, 927,	
	936, 937, 939, 942, 965, 972,	
	973, 975, 983, 998, 1008, 1012,	
	1013, 1018, 1065, 1150, 1151, 1156	
\CurrentOption@SaveLevel	1002, 1005, 1009, 1010, 1015, 1016	
\CurrentOptionKey	778, 781	
\CurrentOptionValue	56, 776, 777, 779, 786	
D		
\DeclareBoolOption	5, 20, 22, 255, 465	
\DeclareComplementaryOption	5, 287, 466	
\DeclareDefaultOption	6, 55, 452	
\DeclareLocalOption	6	
\DeclareLocalOptions	455	
\DeclareOption	223, 224	
\DeclareStringOption	4, 26, 28, 50, 52, 365, 467	
\DeclareVoidOption	6, 32, 33, 34, 401	
\default@ds	968	
\define@key	230, 241,	
	272, 312, 382, 411, 470, 474, 539	
\detokenize	873, 877, 983, 985, 996, 998, 1170	
\DisableKeyvalOption	7, 478	
\do	602, 646, 677, 735, 766, 911,	
	936, 952, 965, 972, 1012, 1150, 1235	
\documentclass	856, 1093	
\ds@	898	
E		
\emph	48, 87, 91, 93	
\empty	110, 111	
\end	1341	
\endcsname	107, 115, 141,	
	157, 164, 232, 238, 243, 249,	
	259, 260, 268, 306, 307, 308,	
	309, 328, 358, 395, 420, 459,	
	489, 497, 501, 513, 517, 540,	
	545, 546, 548, 549, 641, 726,	
	789, 802, 844, 874, 886, 891,	
	893, 905, 927, 942, 973, 989,	
	1005, 1013, 1064, 1078, 1102,	
	1108, 1109, 1120, 1148, 1270,	
	1273, 1276, 1279, 1318, 1340, 1386	
\endinput	123, 839	
\etex@unexpanded	610, 614,	
	1185, 1192, 1221, 1222, 1226, 1227	
\ExecuteOptions	1003	
F		
\futurelet	1195	
G		
\gdef	656, 885, 890, 1024	
I		
\ifetex@unexpanded	851	
\ifin@	925, 940, 958	
\ifKV@dyn@global	495,	
	499, 506, 511, 515, 522, 543, 559	
\ifMCS@print	79	
\ifnum	1306, 1314, 1354, 1362	
\ifpdf	25	
\ifx	56, 88, 108, 111, 115,	
	141, 149, 152, 274, 294, 314,	
	333, 336, 339, 352, 376, 385,	
	413, 433, 435, 445, 532, 552,	
	564, 575, 598, 600, 608, 643,	
	650, 653, 714, 764, 767, 777,	
	789, 844, 912, 915, 934, 937,	
	953, 992, 994, 1021, 1022, 1042,	
	1044, 1124, 1143, 1151, 1170,	
	1180, 1199, 1236, 1238, 1241,	
	1255, 1270, 1273, 1276, 1279, 1318	
\immediate	117, 143	
\in@	922	
\in@false	950	
\in@true	954, 959	
\input	1263, 1319	
\InputIfFileExists	1103	
\iterate	1287, 1289, 1291	

K	
\KV@sp@def	786
\KV@setcurrentvalue	<u>785</u>
\KV@action@error	525
\KV@action@ignore	509
\KV@action@undef	493
\KV@action@warning	528
\KV@AtEnd	177, 178, 794, 815, 816, 837, 838, 848, 858, 866, 1260
\KV@boolkey	273, 313, <u>330</u>
\KV@calldefault	694, 752, <u>760</u>
\KV@checkfirstA	1204, 1207
\KV@checkfirsttok	1195, 1198
\KV@checkkv	1165, 1176
\KV@CurrentOption	602, 604, 607, 615, 619, 624, 628
\KV@DeclareLocalOption ...	456, <u>458</u>
\KV@DeclareStringOption	367, 369, <u>372</u>
\KV@disable@error	563
\KV@disable@warning	574
\KV@do@action	526, 529, 531
\KV@ExecuteOptions	1004, 1007
\KV@false	336, <u>361</u>
\KV@family	230, 272, 296, 312, 382, 411, 459, 589, 705
\KV@fileswith@pti@ns	1020, 1254, 1257
\KV@getkey 603, 606, 648, 679, 737, 759	
\KV@ifdefinable	256, 257, 258, 304, 305, <u>327</u> , 373, 404
\KV@ifempty	1164, 1167, 1169, 1177, 1208, 1211, 1219
\KV@in@	939, 949
\KV@nil	1195, 1218, 1222, 1231
\KV@normalize ..	870, 883, 1023, 1158
\KV@onefilewithoptions	1028, 1033, 1039, 1046, 1060
\KV@param	331, 332, 333, 336, 345, 357, 358
\KV@Patch	608, 650, <u>1259</u>
\KV@prefix	<u>241</u> , 256, 257, 258, 260, 261, 268, 279, 288, 298, 304, 305, 306, 307, 308, 309, 319, 373, 374, 395, 404, 420, 424
\KV@process@pti@ns ...	931, 947, 964
\KV@process@ptions	908, 910
\KV@ProcessKeyvalOptions	589, 593, 595
\KV@ProcessLocalKeyvalOptions .	705, 709, 711
\KV@removeelement	1232
\KV@removegarbage	1222, 1231
\KV@removelastspace	1201, 1209, 1212, 1214, 1218
\KV@result 1159, 1161, 1184, 1185,	1191, 1192, 1220, 1221, 1225, 1226
\KV@SanitizedCurrentOption	966, 989, 1132, 1135, 1155
\KV@setcurrents	769, <u>775</u>
\KV@setcurrentvalue	782, 785
\KV@splitcomma	1160, 1163, 1167
\KV@temp	597, 609, 611, 621, 622, 638, 645, 672, 676, 693, 697, 713, 723, 730, 734, 751, 755, 870, 877, 883, 887, 893, 1023, 1025, 1038, 1046, 1055
\KV@true	333, <u>361</u>
\KV@use@ption 926, 941, 967, 970, 979	
\KV@VOID@	411, 427, 433
\KV@voidkey	412, <u>428</u>
\KV@x	1179, 1180, 1185
\KV@xprocess@ptions	908, 933
\KV@dyn@action	482, 485, 489
\KV@dyn@ext ...	469, 472, 476, 541, 552
\KV@dyn@name 468, 471, 475, 532, 541, 557	
L	
\LoadClass	1124
\LoadCommand	1319, 1336
\loop	1285, 1301, 1312, 1349, 1360
M	
\makeatletter	1067, 1345
\MCS@driver	84
\MCS@emph	88, 94
\meaning ...	205, 485, 1078, 1082, 1156
\MessageBreak	58, 263, 264, 290, 295, 297, 299, 300, 345, 441, 485, 505, 521, 534, 558, 570, 581, 854, 862, 863, 1085, 1086, 1088, 1089, 1090, 1092, 1094, 1117, 1118, 1119, 1120, 1136
N	
\NeedsTeXFormat	4, 797, 1344
\newcommand ..	40, 43, 252, 255, 287, 365, 401, 452, 455, 478, 585, 701
\next ..	403, 405, 408, 1291, 1293, 1295
\numexpr	1010, 1016
O	
\on@line	1116
\OptionNotUsed	991
P	
\PackageError	289, 483, 533, 567, 716, 853
\PackageInfo	120, 355, 388, 448, 504, 520, 555
\PackageWarning ...	262, 342, 438, 578
\PackageWarningNoLine ..	57, 845, 861
\PassOptionsToPackage	63, 80
\ProcessKeyvalOptions 3, 74, 585, 790	
\ProcessLocalKeyvalOptions 4, 701, 717	
\ProcessOptions	229, 897
\protect	213
\ProvidesPackage	5, 112, 158, 841
R	
\RangeCatcodeInvalid	1310, 1322, 1323, 1324, 1325, 1358, 1367, 1368, 1369, 1370
\renewcommand	93
\repeat 1285, 1297, 1308, 1316, 1356, 1364	
\RequirePackage	7, 8, 84, 200, 202, 226, 850, 855, 1384, 1385

\reserved@a	1027, 1032, 1052, 1053, 1056, 1058, 1068, 1128	\toks@	377, 379, 382, 384, 391, 638, 640, 644, 645, 664, 665, 671, 672, 675, 676, 685, 686, 695, 723, 725, 729, 730, 733, 734, 743, 744, 753, 916, 918, 922, 1054, 1055, 1234, 1240, 1243, 1245, 1250
\reserved@b	1041, 1049, 1053	\tw@	636, 681, 682, 694, 721, 739, 740, 752, 920, 922
\RestoreCatcodes	1299, 1302, 1303, 1337, 1347, 1350, 1351, 1382		
S			
\setkeys	44, 253, 480, 695, 753		
\SetupDriver	32, 33, 34, 35, 38, 40		
\SetupKeyvalOptions . .	4, 13, 252, 461		
\space	294, 1083, 1086, 1089, 1091, 1095, 1118, 1133, 1136, 1138, 1139		
\strip@prefix	205, 209, 485, 1077, 1082, 1156		
T			
\t	1240, 1241		
\Test	1321, 1339, 1366, 1384, 1385		
\textcolor	94		
\the	165, 166, 167, 168, 179, 382, 391, 644, 665, 672, 675, 682, 686, 694, 695, 730, 733, 740, 744, 752, 753, 803, 804, 805, 806, 817, 922, 1010, 1016, 1055, 1240, 1243, 1245, 1250, 1304, 1352		
\TMP@EnsureCode . . .	176, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 814, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836		
\toks	636, 681, 682, 694, 721, 739, 740, 752, 920, 922		
U			
\unexpanded	854		
W			
\write	117, 143		
X			
\x	107, 108, 111, 116, 120, 122, 142, 147, 157, 163, 171, 271, 282, 311, 322, 381, 398, 410, 423, 431, 433, 481, 491, 494, 510, 538, 588, 591, 704, 707, 762, 764, 801, 809, 871, 880, 921, 924, 952, 953, 957, 962, 981, 988, 1074, 1076, 1086, 1092, 1235, 1236, 1238, 1245, 1249, 1252		
Y			
\y	1081, 1082, 1089, 1092		
Z			
\zap@space	781, 1052, 1179, 1193		