

# The `l3keys2e` package

## $\text{\LaTeX} 2_{\epsilon}$ option processing using $\text{\LaTeX} 3$ keys\*

The  $\text{\LaTeX} 3$  Project<sup>†</sup>

Released 2013/10/08

The key–value method for optional arguments is very popular, as it allows the class or package author to define a large number of options with a simple interface. The `expl3` bundle of  $\text{\LaTeX} 3$  base code includes the module `l3keys` for defining keys, but to use these when loading  $\text{\LaTeX} 2_{\epsilon}$  packages and classes requires extra support. That support is provided by this small package, which is intended to enable  $\text{\LaTeX} 2_{\epsilon}$  packages to benefit from the power of the  $\text{\LaTeX} 3$  key–value system.

### 0.1 Creating and using keyval options

As with any key–value input, using key–value pairs as package or class options has two parts. The first stage is to define one or more keys, using the `\keys_define:nn` function. For example, an option which simply stores a value would be created using:

```
\keys_define:nn { module }  
  { option .set:N = \l_module_variable_tl }
```

On its own, this will not make the key an option for the package or class containing the definition. The second stage is therefore to process the current options, searching for applicable keys.

---

**`\ProcessKeysOptions`**

```
\ProcessKeysOptions {<module>}
```

The `\ProcessKeysOptions` function is used to check the current option list against the keys defined for `{<module>}`. Global (class) options and local (package) options are checked when this function is called in a package. Each option which does match a key name is then used to attempt to set the appropriate key using `\keys_set:nn`. For example, the option defined earlier would be processed by the line

```
\ProcessKeysOptions { module }
```

---

\*This file describes v4595, last revised 2013/10/08.

<sup>†</sup>E-mail: [latex-team@latex-project.org](mailto:latex-team@latex-project.org)

<hr/> <code>\ProcessKeysPackageOptions</code> <hr/>	<code>\ProcessKeysPackageOptions {(module)}</code>
	This function works in a similar manner to <code>\ProcessKeysOptions</code> . When used in a $\text{\LaTeX 2}_\epsilon$ package, <code>\ProcessKeysPackageOptions</code> will not examine any class options available. In contrast, <code>\ProcessKeysOptions</code> does parse class options (in common with the $\text{\LaTeX 2}_\epsilon$ kernel function <code>\ProcessOptions</code> ).

## 0.2 l3keys2e Implementation

	<pre> 1 &lt;*package&gt; 2 &lt;@@=keys&gt; 3 \ProvidesExplPackage 4   {\ExplFileName}{\ExplFileDate}{\ExplFileVersion}{\ExplFileDescription} </pre>
<code>\l__keys_latex_options_clist</code>	<p>A single list is used for all options, into which they are collected before processing.</p> <pre> 5 \clist_new:N \l__keys_latex_options_clist </pre> <p><i>(End definition for <code>\l__keys_latex_options_clist</code>. This function is documented on page ??.)</i></p>
<code>\l__keys_process_class_bool</code>	<p>A flag to indicate that class options should be processed for packages.</p> <pre> 6 \bool_new:N \l__keys_process_class_bool </pre> <p><i>(End definition for <code>\l__keys_process_class_bool</code>. This function is documented on page ??.)</i></p>
<code>\__keys_latex_options:n</code>	<p>The main function calls functions to collect up the global and local options into <code>\l__keys_latex_options_clist</code> before calling the underlying functions to actually do the processing. So that a suitable message is produced if the option is unknown, the special <code>unknown</code> key is set if it does not already exist for the current module.</p> <pre> 7 \cs_new_protected:Npn \__keys_latex_options:n #1 8   { 9     \clist_clear:N \l__keys_latex_options_clist 10    \__keys_latex_options_global:n {#1} 11    \__keys_latex_options_local: 12    \keys_if_exist:nnF {#1} { unknown } 13    { 14      \keys_define:nn {#1} 15      { 16        unknown .code:n = 17        { 18          \msg_error:nxxx { keyvalue } { option-unknown } 19          { \l_keys_key_tl } { \@currname } 20        } 21      } 22    } 23    \keys_set:nV {#1} \l__keys_latex_options_clist 24    \AtEndOfPackage { \cs_set_eq:NN \@unprocessedoptions \scan_stop: } 25  } </pre> <p><i>(End definition for <code>\__keys_latex_options:n</code>. This function is documented on page ??.)</i></p>

`\_keys_latex_options_global:n` Global (class) options are handled differently for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> packages and classes. Hence this function is essentially a check on the current file type. The initial test is needed as L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> allows variables to be equal to `\scan_stop:`, which is forbidden in L<sup>A</sup>T<sub>E</sub>X 3 code.

```

26 \cs_new_protected:Npn \_keys_latex_options_global:n #1
27 {
28   \cs_if_eq:NNF \@classoptionslist \scan_stop:
29   {
30     \cs_if_eq:NNTF \@currentt \clsextension
31     { \_keys_latex_options_class:n {#1} }
32     {
33       \bool_if:NT \l__keys_process_class_bool
34       { \_keys_latex_options_package:n {#1} }
35     }
36   }
37 }

```

(End definition for `\_keys_latex_options_global:n`. This function is documented on page ??.)

`\_keys_latex_options_class:n` For classes, each option (stripped of any content after =) is checked for existence as a key. If found, the option is added to the combined list for processing. On the other hand, unused options are stored up in `\@unusedoptionlist`. Before any of that, though, there is a simple check to see if there is an unknown key. If there is, then *everything* will match and the mapping can be skipped.

```

38 \cs_new_protected:Npn \_keys_latex_options_class:n #1
39 {
40   \keys_if_exist:nnTF {#1} { unknown }
41   { \clist_put_right:No \l__keys_latex_options_clist \@classoptionslist }
42   {
43     \clist_map_inline:Nn \@classoptionslist
44     {
45       \keys_if_exist:nnTF {#1} { \_keys_latex_remove_equals:n {##1} }
46       { \clist_put_right:Nn \l__keys_latex_options_clist {##1} }
47       { \clist_put_right:Nn \@unusedoptionlist {##1} }
48     }
49   }
50 }

```

(End definition for `\_keys_latex_options_class:n`. This function is documented on page ??.)

`\_keys_latex_options_package:n` For global options when processing a package, the tasks are slightly different from those for a class. The check is the same, but here there is nothing to do if the option is not applicable. Each valid option also needs to be removed from `\@unusedoptionlist`.

```

51 \cs_new_protected:Npn \_keys_latex_options_package:n #1
52 {
53   \clist_map_inline:Nn \@classoptionslist
54   {
55     \keys_if_exist:nnT {#1} { \_keys_latex_remove_equals:n {##1} }
56     {
57       \clist_put_right:Nn \l__keys_latex_options_clist {##1}
58       \clist_remove_all:Nn \@unusedoptionlist {##1}

```

```

59     }
60   }
61 }

```

(End definition for `\_keys_latex_options_package:n`. This function is documented on page ??.)

`\_keys_latex_options_local:` If local options are found, they are added to the processing list. L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> stores options for each file in a macro which may or may not exist, hence the need to use `\cs_if_exist:c`.

```

62 \cs_new_protected_nopar:Npn \_keys_latex_options_local:
63 {
64   \cs_if_eq:NNF \@currentx \clsextension
65   {
66     \cs_if_exist:cT { opt@ \@currname . \@currentx }
67     {
68       \exp_args:Nnc \clist_put_right:NV \l__keys_latex_options_clist
69       { opt@ \@currname . \@currentx }
70     }
71   }
72 }

```

(End definition for `\_keys_latex_options_local:`. This function is documented on page ??.)

`\_keys_latex_remove_equals:n` As the name suggests, this is a simple function to remove an equals sign from the input.  
`\_keys_latex_remove_equals:w` This is all wrapped up in an `n` function so that there will always be a sign available.

```

73 \cs_new:Npn \_keys_latex_remove_equals:n #1
74 { \_keys_latex_remove_equals:w #1 = \q_stop }
75 \cs_new:Npn \_keys_latex_remove_equals:w #1 = #2 \q_stop {#1}

```

(End definition for `\_keys_latex_remove_equals:n`. This function is documented on page ??.)

**`\ProcessKeysOptions`** The user macro are simply wrappers around the internal process. In contrast to other similar packages, the module name is always required here.

```

76 \cs_new_protected_nopar:Npn \ProcessKeysOptions #1
77 {
78   \bool_set_true:N \l__keys_process_class_bool
79   \_keys_latex_options:n {#1}
80 }
81 \cs_new_protected_nopar:Npn \ProcessKeysPackageOptions #1
82 {
83   \bool_set_false:N \l__keys_process_class_bool
84   \_keys_latex_options:n {#1}
85 }

```

```

86 \@onlypreamble \ProcessKeysOptions
87 \@onlypreamble \ProcessKeysPackageOptions

```

(End definition for `\ProcessKeysOptions`. This function is documented on page 1.)

One message to give.

```

88 \msg_new:nnnn { keyvalue } { option-unknown }
89 { Unknown~option~'~#1'~for~package~#2. }
90 {
91   LaTeX~has~been~asked~to~set~an~option~called~'~#1'~
92   but~the~#2~package~has~not~created~an~option~with~this~name.
93 }

```

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
<code>\@classoptionslist</code> . . . . .	28, 41, 43, 53
<code>\@clsextension</code> . . . . .	30, 64
<code>\@currentx</code> . . . . .	30, 64, 66, 69
<code>\@currname</code> . . . . .	19, 66, 69
<code>\@onlypreamble</code> . . . . .	86, 87
<code>\@unprocessedoptions</code> . . . . .	24
<code>\@unusedoptionlist</code> . . . . .	47, 58
<code>\__keys_latex_options:n</code> . . . .	7, 7, 79, 84
<code>\__keys_latex_options_class:n</code>	31, 38, 38
<code>\__keys_latex_options_global:n</code> . . . .	10, 26, 26
<code>\__keys_latex_options_local:</code>	11, 62, 62
<code>\__keys_latex_options_package:n</code> . . . .	34, 51, 51
<code>\__keys_latex_remove_equals:n</code> . . . .	45, 55, 73, 73
<code>\__keys_latex_remove_equals:w</code>	73, 74, 75
A	
<code>\AtEndOfPackage</code> . . . . .	24
B	
<code>\bool_if:NT</code> . . . . .	33
<code>\bool_new:N</code> . . . . .	6
<code>\bool_set_false:N</code> . . . . .	83
<code>\bool_set_true:N</code> . . . . .	78
C	
<code>\clist_clear:N</code> . . . . .	9
<code>\clist_map_inline:Nn</code> . . . . .	43, 53
<code>\clist_new:N</code> . . . . .	5
<code>\clist_put_right:Nn</code> . . . . .	46, 47, 57
<code>\clist_put_right:No</code> . . . . .	41
<code>\clist_put_right:Nv</code> . . . . .	68
<code>\clist_remove_all:Nn</code> . . . . .	58
<code>\cs_if_eq:NNF</code> . . . . .	28, 64
<code>\cs_if_eq:NNTF</code> . . . . .	30
<code>\cs_if_exist:cT</code> . . . . .	66
<code>\cs_new:Npn</code> . . . . .	73, 75
<code>\cs_new_protected:Npn</code> . . . . .	7, 26, 38, 51
<code>\cs_new_protected_nopar:Npn</code> . . . .	62, 76, 81
<code>\cs_set_eq:NN</code> . . . . .	24
E	
<code>\exp_args:NNc</code> . . . . .	68
<code>\ExplFileDate</code> . . . . .	4
<code>\ExplFileDescription</code> . . . . .	4
<code>\ExplFileName</code> . . . . .	4
<code>\ExplFileVersion</code> . . . . .	4
K	
<code>\keys_define:nn</code> . . . . .	14
<code>\keys_if_exist:nnF</code> . . . . .	12
<code>\keys_if_exist:nnT</code> . . . . .	55
<code>\keys_if_exist:nnTF</code> . . . . .	40, 45
<code>\keys_set:nV</code> . . . . .	23
L	
<code>\l__keys_latex_options_clist</code> . . . . .	5, 5, 9, 23, 41, 46, 57, 68
<code>\l__keys_process_class_bool</code> . . . . .	6, 6, 33, 78, 83
<code>\l_keys_key_tl</code> . . . . .	19
M	
<code>\msg_error:nnxx</code> . . . . .	18
<code>\msg_new:nnnn</code> . . . . .	88
P	
<code>\ProcessKeysOptions</code> . . . . .	1, 76, 76, 76, 86
<code>\ProcessKeysPackageOptions</code> . . . .	2, 81, 87
<code>\ProvidesExplPackage</code> . . . . .	3
Q	
<code>\q_stop</code> . . . . .	74, 75
S	
<code>\scan_stop:</code> . . . . .	24, 28