

# The latexrelease package<sup>\*</sup>

The L<sup>A</sup>T<sub>E</sub>X Project

2025/01/31

This file is maintained by the L<sup>A</sup>T<sub>E</sub>X Project team.  
Bug reports can be opened (category `latex`) at  
<https://latex-project.org/bugs.html>.

## 1 Introduction

Prior to the 2015 release of L<sup>A</sup>T<sub>E</sub>X, essentially no changes had been made to the L<sup>A</sup>T<sub>E</sub>X format code for some years, with all improvements being instead added to the package `fixltx2e`.

While this worked at a technical level it meant that you had to explicitly opt-in to bug fixes and improvements, and the vast majority of documents did not benefit.

As described in L<sup>A</sup>T<sub>E</sub>X News 22, a new policy is being implemented in which improvements will now be added to the format by default, and this `latexrelease` package may be used to ensure stability where needed, either by making a new format use an older definition of some commands, or conversely may be used to supply the new definitions for use with an old format.

The basic use is:

```
\RequirePackage[2015/01/01]{latexrelease}  
\documentclass{article}  
....
```

After such a declaration the document will use definitions current in the January 2015 L<sup>A</sup>T<sub>E</sub>X, whether the actual format being used is older, or newer than that date. In the former case a copy of `latexrelease.sty` would need to be made available for use with the older format. This may be used, for example, to share a document between co-workers using different L<sup>A</sup>T<sub>E</sub>X releases, or to protect a document from being affected by system updates. As well as the definitions within the format itself, individual packages may use the commands defined here to adjust their definitions to the specified date as described below.

Note that the `latexrelease` package is intended for use at the start of a *document*. Package and class code should not include this package as loading a package should not normally globally reset the effective version of L<sup>A</sup>T<sub>E</sub>X that is in force, so affecting all other packages used in the document.

---

<sup>\*</sup>This file has version number v1.0q, last revised 2025/01/31.

The bulk of this package, after some initial setup and option handling consists of a series of `\IncludeInRelease` commands which have been extracted from the main source files of the `LATEX` format. These contain the old and new versions of any commands with modified definitions.

## 2 Package Options

- `yyyy/mm/dd` or `yyyy-nn-dd` The package accepts any possible `LATEX` format date as argument, although dates in the future for which the current release of this package has no information will generate a warning. Dates earlier than 2015 will work but will roll back to some point in 2015 when the method was introduced. The `\requestedLaTeXdate` is set to the normalized date argument so that package rollback defaults to the specified date.
- `current` This is the default behaviour, it does not change the effective date of the format but does ensure that the `\IncludeInRelease` command is defined. The `\requestedLaTeXdate` macro is reset to 0 so that package rollback does not use the implicit date.
- `latest` sets the effective date of the format to the release date of this file, so in an older format applies all patches currently available. The `\requestedLaTeXdate` macro is reset to 0 so that package rollback does not use the implicit date.

In all cases, when the package is loaded, the `\sourceLaTeXdate` is defined to be the numerical representation of `\fmtversion` before the rollback/forward actually happens, so it is possible to test from which was the original `LATEX` version before `latexrelease` was loaded. This is particularly useful when some code in a package has to be redefined differently if rolling backwards in time or forwards.

## 3 Release Specific Code

The `\IncludeInRelease` mechanism allows the kernel developer to associate code with a specific date to choose different versions of definitions depending on the date specified as an option to the `latexrelease` package. Is also available for use by package authors (or even in a document if necessary).

`\IncludeInRelease` `{<code-date>}[<format-date>]{<label>}{<message>}<code>\EndIncludeInRelease`

`{<code-date>}` This date is associated with the `{<code>}` argument and will be compared to the requested date in the option to the `latexrelease`.

`[<format-date>]` This optional argument can be used to specify a format date with the code in addition to the mandatory `{<code-date>}` argument. This can be useful for package developers as described below.

`{<label>}` The `{<label>}` argument is an identifier (string) that within a given package must be a unique label for each related set of optional definitions. Per package at most one code block from all the `\IncludeInRelease` declarations with the same label will be executed.

`{\message}` The `{\message}` is an informative string that is used in messages. It has no other function.

`<code>` Any  $\text{\TeX}$  code after the `\IncludeInRelease` arguments up until the and the following `\EndIncludeInRelease` is to be conditionally included depending on the date of the format as described below.

The `\IncludeInRelease` declarations with a given label should be in reverse chronological order in the file. The one chosen will depend on this order, the effective format version and the date options, as described below.

If your package `mypackage` defines a `\widget` command but has one definition using the features available in the 2015  $\text{\LaTeX}$  release, and a different definition is required for older formats then you can use:

```
\IncludeInRelease{2015/01/01}{\widget}{Widget Definition}
\def\widget{new version}%
\EndIncludeInRelease

\IncludeInRelease{0000/00/00}{\widget}{Widget Definition}
\def\widget{old version}%
\EndIncludeInRelease
```

If a document using this package is used with a format with effective release date of 2015/01/01 or later the new code will be used, otherwise the old code will be used. Note the *effective release date* might be the original  $\text{\LaTeX}$  release date as shown at the start of every  $\text{\LaTeX}$  job, or it may be set by the `latexrelease` package, so for example a document author who wants to ensure the new version is used could use

```
\RequirePackage[2015/01/01]{latexrelease}
\documentclass{article}
\usepackage{mypackage}
```

If the document is used with a  $\text{\LaTeX}$  format from 2014 or before, then `latexrelease` will not have been part of the original distribution, but it may be obtained from a later  $\text{\LaTeX}$  release or from CTAN and distributed with the document, it will make an older  $\text{\LaTeX}$  release act essentially like the 2015 release.

### 3.1 Intermediate Package Releases

The above example works well for testing against the `latex` format but is not always ideal for controlling code by the release date of the *package*. Suppose  $\text{\LaTeX}$  is not updated but in March you update the `mypackage` package and modify the definition of `\widget`. You could code the package as:

```
\IncludeInRelease{2015/03/01}{\widget}{Widget Definition}
\def\widget{even newer improved March version}%
\EndIncludeInRelease

\IncludeInRelease{2015/01/01}{\widget}{Widget Definition}
\def\widget{new version}%
\EndIncludeInRelease

\IncludeInRelease{0000/00/00}{\widget}{Widget Definition}
```

```
\def\widget{old version}%
\EndIncludeInRelease
```

This would work and allow a document author to choose a date such as

```
\RequirePackage[2015/03/01]{latexrelease}
\documentclass{article}
\usepackage{mypackage}
```

To use the latest version, however it would have disadvantage that until the next release of L<sup>A</sup>T<sub>E</sub>X, by default, if the document does not use `latexrelease` to specify a date, the new improved code will not be selected as the effective date will be 2015/01/01 and so the first code block will be skipped.

For this reason `\IncludeInRelease` has an optional argument that specifies an alternative date to use if a date option has not been specified to `latexrelease`.

```
\IncludeInRelease{2015/03/01}[2015/01/01]{\widget}{Widget Definition}
\def\widget{even newer improved March version}%
\EndIncludeInRelease
```

```
\IncludeInRelease{2015/01/01}{\widget}{Widget Definition}
\def\widget{new version}%
\EndIncludeInRelease
```

```
\IncludeInRelease{0000/00/00}{\widget}{Widget Definition}
\def\widget{old version}%
\EndIncludeInRelease
```

Now, by default on a 2015/01/01 L<sup>A</sup>T<sub>E</sub>X format, the first code block will compare the format date to the optional argument 2015/01/01 and so will execute the *even newer improved* version. The remaining blocks using the `\widget` label argument will all then be skipped.

If on the other hand the document requests an explicit release date using `latexrelease` then this date will be used to decide what code block to include.

## 3.2 Using `\IncludeInRelease` in Packages

If `\IncludeInRelease` is used within a package then all such conditional code needs to be within such declarations, e.g., it is not possible in the above example to have the “current” definition of `\widget` somewhere in the main code and only the two older definitions inside `\IncludeInRelease` declarations. If you would do this then one of those `\IncludeInRelease` declarations would be included overwriting the even newer code in the main part of the package. As a result your package may get fragmented over time with various `\IncludeInRelease` declarations sprinkled throughout your code or you have to interrupt the reading flow by putting those declarations together but not necessarily in the place where they belong.

To avoid this issue you can use the following coding strategy: place the current `\widget` definition in the main code where it correctly belongs.

```
...
\def\widget {even newer improved March version}
\def\@widget{newly added helper command no defined in older releases}
...
```

Then, near the end of your package place the following:

```
\IncludeInRelease{2015/03/01}[2015/01/01]{\widget}{Widget Definition}
\EndIncludeInRelease

\IncludeInRelease{2015/01/01}{\widget}{Widget Definition}
\def\widget{new version}%
\let\@widget\@undefined % this doesn't exist in earlier releases
\EndIncludeInRelease

\IncludeInRelease{0000/00/00}{\widget}{Widget Definition}
\def\widget{old version}%
\EndIncludeInRelease
```

This way the empty code block hides the other `\IncludeInRelease` declarations unless there is an explicit request with a date 2015/01/01 or earlier.

Now if you make a further change to `\widget` in the future you simply copy the current definition into the empty block and add a new empty declaration with today's date and the current format date. This way your main code stays readable and the old versions accumulate at the end of the package.<sup>1</sup>

The only other “extra effort” necessary when using this approach is that it may be advisable to undo new definitions in the code block for the previous release, e.g., in the above example we undefined `\@widget` as that isn't available in the 2015/01/01 release but was defined in the main code. If all your conditional code is within `\IncludeInRelease` declarations that wouldn't been necessary as the new code only gets defined if that release is chosen.

## 4 Declaring entire modules

Sometimes a large chunk of code is added as a module to another larger code base. As example of that in the 2020-10-01 release L<sup>A</sup>T<sub>E</sub>X got a new hook management system, `lthooks`, which was added in one go and, as with all changes to the kernel, it was added to `latexrelease`. However rolling back from a future date to the 2020-10-01 release didn't work because `latexrelease` would try to define again all those commands, which would result in many “already defined” errors and similar issues.

To solve that problem, completely new modules can be defined in `latexrelease` using the commands:

```
\NewModuleRelease{<initial release date>}{<name>}{<message>}
  <module code>
\IncludeInRelease{0000/00/00}{<name>}{<message>}
  <undefine module code>
\EndModuleRelease
```

With that setup, the module `<name>` will be declared to exist only in releases equal or later `<initial release date>`.

---

<sup>1</sup>Of course there may be some cases in which the old code has to be in a specific place within the package as other code depends on it (e.g., if you `\let` something to it). In that case you have to place the code variations in the right place in your package rather than accumulating them at the very end.

If `latexrelease` is rolling backwards or forwards between dates after *⟨initial release date⟩*, then all the *⟨module code⟩* is skipped, except when inside *⟨IncludeInRelease⟩* guards, in which case the code is applied or skipped as discussed above.

If rolling forward from a date before the module’s *⟨initial release date⟩* to a date after that, then all the *⟨module code⟩* is executed to define the module, and `\IncludeInRelease` guards are executed accordingly, depending on the date declared and the target date.

If `latexrelease` is rolling back to a date before *⟨release date⟩*, then the code in the `\IncludeInRelease` guard dated 0000/00/00 is executed instead to undefine the module. This guard *is not* ended by the usual `\EndIncludeInRelease`, but instead by `\EndModuleRelease`.

Finally, if rolling backwards or forwards between dates both before *⟨initial release date⟩*, the entire code between *⟨NewModuleRelease⟩* and *⟨EndModuleRelease⟩* is entirely skipped.

## 4.1 Example

Here is an example usage of the structure described above, as it would be used in the L<sup>A</sup>T<sub>E</sub>X kernel, taking `lhooks` as example:

```
%<*2ekernel|latexrelease>
\ExplSyntaxOn
%<latexrelease>\NewModuleRelease{2020/10/01}{lhooks}%
%<latexrelease>          {The~hook~management~system}
\NewDocumentCommand \NewHook { m }
  { \hook_new:n {#1} }
%<latexrelease>\IncludeInRelease{2021/06/01}{\AddToHook}{Long~argument}
\NewDocumentCommand \AddToHook { m o +m }
  { \hook_gput_code:nnn {#1} {#2} {#3} }
%<latexrelease>\EndIncludeInRelease
%<latexrelease>
%<latexrelease>\IncludeInRelease{2020/10/01}{\AddToHook}{Long~argument}
%<latexrelease>\NewDocumentCommand \AddToHook { m o m }
%<latexrelease> { \hook_gput_code:nnn {#1} {#2} {#3} }
%<latexrelease>\EndIncludeInRelease
%<latexrelease>
%<latexrelease>\IncludeInRelease{0000/00/00}{lhooks}{Undefine~lhooks}
%<latexrelease>\cs_undefine:N \NewHook
%<latexrelease>\cs_undefine:N \AddToHook
%<latexrelease>\EndModuleRelease
\ExplSyntaxOff
%</2ekernel|latexrelease>
```

In the example above, `\NewHook` is declared only once, and unchanged in the next release (2021/06/01 in the example), so it has no `\IncludeInRelease` guards, and will only be defined if needed. `\AddToHook`, on the other hand, changed between the two releases (made up for the example; it didn’t really happen) and has an `\IncludeInRelease` block for the current release (off `docstrip` guards, so it goes into the kernel too), and another for the previous release (in `docstrip` guards so it goes only into `latexrelease`).

Note that in the example above, `\ExplSyntaxOn` and `\ExplSyntaxOff` were added *outside* the module code because, as discussed above, sometimes the code

outside `\IncludeInRelease` guards may be skipped, but not the code inside them, and in that case the catcodes would be wrong when defining the code.

## 5 fixltx2e

As noted above, prior to the 2015 L<sup>A</sup>T<sub>E</sub>X release updates to the L<sup>A</sup>T<sub>E</sub>X kernel were not made in the format source files but were made available in the `fixltx2e` package. That package is no longer needed but we generate a small package from this source that just makes a warning message but otherwise does nothing.

## 6 Implementation

We require at least a somewhat sane version of L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. Earlier ones where really quite different from one another.

```
1 (*latexrelease)
2 \NeedsTeXFormat{LaTeX2e}[1996/06/01]
```

### 6.1 Setup

`\sourceLaTeXdate` Store the original L<sup>A</sup>T<sub>E</sub>X format version as a number in the format `YYYYMMDD`. This macro has to be defined conditionally, so that it isn't changed in case `latexrelease.sty` is reloaded, but it can't be defined in the kernel only, otherwise `latexrelease.sty` wouldn't work in older L<sup>A</sup>T<sub>E</sub>X due to the missing macro.

```
3 \@ifundefined{sourceLaTeXdate}{%
4   \edef\sourceLaTeXdate{%
5     \expandafter\@parse@version\fmtversion//00\@nil}}{}
```

`\IncludeInRelease` These are defined in `ltvers.dtx`.

`\EndIncludeInRelease`

```
6 \DeclareOption*{%
7   \def\@IncludeInRelease#1[#2]{\@IncludeInRelease{#1}}%
8   \let\requestedpatchdate\CurrentOption}
9 \DeclareOption{latest}{%
10  \let\requestedpatchdate\latexreleaseversion
11  \AtEndOfPackage{\def\requestedLaTeXdate{0}}}%
12 \DeclareOption{current}{%
13  \let\requestedpatchdate\fmtversion
14  \AtEndOfPackage{\def\requestedLaTeXdate{0}}}%
15 \let\requestedpatchdate\fmtversion
16 \ProcessOptions\relax
```

Sanity check options, it allows some non-legal dates but always ensures `requestedLaTeXdate` gets set to a number. Generate an error if there are any non digit tokens remaining after removing the `//`.

```
17 \def\reserved@a{%
18  \edef\requestedLaTeXdate{\the\count@}%
19  \reserved@b}
20 \def\reserved@b#1\\{%
21  \def\reserved@b{#1}}%
22 \ifx\reserved@b\empty\else
23  \PackageError{latexrelease}%
```

```

24             {Unexpected option \requestedpatchdate}%
25             {The option must be of the form yyyy/mm/dd or yyyy-mm-dd}%
26 \fi}
27 \afterassignment\reserved@a
28 \count@\expandafter
29 \@parse@version\expandafter0\requestedpatchdate//00\@nil\
    less precautions needed for \fmtversion
30 \edef\currentLaTeXdate{%
31 \expandafter\@parse@version\fmtversion//00\@nil}
32 \ifnum\requestedLaTeXdate=\currentLaTeXdate
33 \PackageWarningNoLine{latexrelease}{%
34 Current format date selected, no patches applied}
35 \expandafter\endinput
36 \fi

```

A newer version of latexrelease should have been distributed with the later format.

```

37 \ifnum\currentLaTeXdate
38 >\expandafter\@parse@version\latexreleaseversion//00\@nil
39 \PackageWarningNoLine{latexrelease}{%
40 The current package is for an older LaTeX format:\MessageBreak
41 LaTeX \latexreleaseversion\space\MessageBreak
42 Obtain a newer version of this package!}
43 \expandafter\endinput
44 \fi

```

can't patch into the future, could make this an error but it has some uses to control package updates so allow for now.

```

45 \ifnum\requestedLaTeXdate
46 >\expandafter\@parse@version\latexreleaseversion//00\@nil
47 \PackageWarningNoLine{latexrelease}{%
48 The current package is for LaTeX \latexreleaseversion:\MessageBreak
49 It has no patches beyond that date\MessageBreak
50 There may be an updated version\MessageBreak
51 of this package available from CTAN}
52 \expandafter\endinput
53 \fi

```

Update the format version to the requested date.

```

54 \let\fmtversion\requestedpatchdate
55 \let\currentLaTeXdate\requestedLaTeXdate

```

## 6.2 Ignoring \_new errors when rolling back

Enforce \ExplSyntaxOn and \ExplSyntaxOff to be \relax in latexrelease if they are not yet defined. They are later restored to be undefined if needed.

```

56 \csname ExplSyntaxOn\endcsname
57 \csname ExplSyntaxOff\endcsname

```

Define a set of changes here, but we'll only use them later to make sure they are applied after expl3 is loaded. If loading from a rather old format, we don't have \ExplSyntaxOn yet.

```

58 \begingroup
59 \endlinechar=-1

```



```

60 \catcode95=11 % _
61 \catcode58=11 % :
62 \catcode126=10 % ~
63 \catcode32=09 % <space>
64 \xdef\latexrelease@postltxpl{\unexpanded{%
65 (==latexrelease)

```

First we'll define a `\declarecommand` that does `\renewcommand` if the command being defined already exists, and `\newcommand` otherwise.

```

66 \cs_gset_protected:Npn \@@_declare_command:w
67 { \@star@or@long \@@_declare_command:Nw }
68 \cs_gset_protected:Npn \@@_declare_command:Nw #1
69 { \cs_if_exist:NTF #1 { \renewcommand } { \newcommand } #1 }

```

Then define a version of `\e@alloc` that checks if the control sequence being defined already exists, and if so, checks if its meaning is the same as the one that would be defined with the call to `\e@alloc`. If both tests pass, nothing is defined to save a register. This version also takes care of setting `\allocationnumber` to the value it would have after the register is allocated.

```

70 \cs_gset_protected:Npn \@@_e@alloc:NnnnnN #1 #2 #3 #4 #5 #6
71 {
72   \cs_if_free:NTF #6
73   { \use:n }
74   {
75     \exp_after:wN \@@_e@alloc:N
76     \token_to_meaning:N #6 \scan_stop: {#2} #6
77   }
78   { \@@_e@alloc #1 {#2} {#3} {#4} {#5} #6 }
79 }

```

Walk through the meaning of the control sequence token by token, looking for the register allocation number.

```

80 \cs_gset_protected:Npn \@@_e@alloc:N #1
81 {
82   \if_int_compare:w 0 < 0
83   \if_int_compare:w 10 < 9#1 ~ 1 \fi:
84   \if_charcode:w " #1 1 \fi: \exp_stop_f:
85   \tex_afterassignment:D \@@_e@alloc:w
86   \@tempcnta #1
87   \use_i:nnn
88   \fi:
89   \use:n
90   {
91     \if_meaning:w \scan_stop: #1
92     \exp_after:wN \use_iv:nnnn
93     \fi:
94     \@@_e@alloc:N
95   }
96 }

```

When found, check if it is the exact same register as it would be allocated, and if it is, set `\allocationnumber` accordingly and exit, otherwise undefine the register and allocate from scratch.

```

97 \cs_gset_protected:Npn \@@_e@alloc:w #1 \scan_stop: #2 #3
98 {

```

```

99      #2 \@@_tmp:w = \@tempcnta
100     \token_if_eq_meaning:NNTF #3 \@@_tmp:w
101     { \int_set_eq:NN \allocationnumber \@tempcnta \use_none:n }
102     { \cs_set_eq:NN #3 \tex_undefined:D \use:n }
103 }

```

Now create a token list to hold the list of changed commands, and define a temporary macro that will loop through the command list, store each in `\l_@@_restores_tl`, save a copy, and redefine each.

```

104 \tl_clear_new:N \l_@@_restores_tl
105 \cs_gset:Npn \@@_redefines:w #1 #2
106 {
107   \quark_if_recursion_tail_stop:N #1
108   \tl_put_right:Nn \l_@@_restores_tl {#1}
109   \cs_set_eq:cN { @@_ \cs_to_str:N #1 } #1
110   \cs_set_eq:NN #1 #2
111   \@@_redefines:w
112 }

```

The redefinitions below are needed because:

`\__kernel_chk_if_free_cs:N` This function is used ubiquitously in the `l3kernel` to check if a control sequence is definable, and give an error otherwise (similar to `\@ifdefinable`). Making it a no-op is enough for most cases (except when defining new registers);

`\e@alloc` In the case of new registers, we waste an allocation number if we do `\new\meta {thing}` in a register that's already allocated, so the redefinition of `\e@alloc` checks if the new register is really necessary. This code does not clear the register, which might cause problems in the future, if a register is allocated but not properly cleared before using;

`\__kernel_msg_error:nxx` This command is used to error on already defined scan marks. Just making the error do nothing is enough, as no action is taken in that case;

`\msg_new:nnnn` Used to define new messages. Making it `_gset` is enough. Other msg commands like `\msg_new:nnn` and `\__kernel_msg_new:nnn(n)` are defined in terms of `\msg_new:nnnn`, so there is no need to change the other ones;

`\NewDocumentCommand` Used to define user-level commands in the kernel. Making it equal to `\DeclareDocumentCommand` solves the problem;

`\newcommand` Same as above.

And here we go:

```

113 \@@_redefines:w
114   \__kernel_chk_if_free_cs:N \use_none:n
115   \e@alloc \@@_e@alloc:NnnnnN
116   \__kernel_msg_error:nxx \use_none:nnn
117   \msg_new:nnnn \msg_gset:nnnn
118   % \NewDocumentCommand \DeclareDocumentCommand % after ltcmd.dtx
119   \newcommand \@@_declare_command:w

```

Temp addition ...

```
120 \_kernel_msg_error:nnn \use_none:nnn % needed while redirect for kernel msgs doesn't work
121 \q_recursion_tail \q_recursion_tail
122 \q_recursion_stop
```

Finally, redirect the error thrown by `\NewHook` to nowhere so it can be safely reused (the hook isn't redeclared if it already exists). The same happens for `\NewMarkClass`.

```
123 \msg_redirect_name:nnn { hooks } { exists } { none }
124 \msg_redirect_name:nnn { mark } { class-already-defined } { none }
```

The same is also needed for `\NewSocket` and `\NewSocketPlug`.

```
125 \msg_redirect_name:nnn { socket } { already-declared } { none }
126 \msg_redirect_name:nnn { socket } { plug-already-declared } { none }
```

Now a one-off for `ltxcmd.dtx`: we need to make `\NewDocumentCommand` not complain on an already existing command, but it has to be done after `\NewDocumentCommand` is defined, so this is separate from the `\latexrelease@postltxexpl` actions above:

```
127 \cs_gset_protected:Npn \latexrelease@postltxcmd
128 {
129   \@@_redefines:w
130     \NewDocumentCommand \DeclareDocumentCommand
131     \q_recursion_tail \q_recursion_tail
132     \q_recursion_stop
133 }
134 }%
135 \endgroup
136 \</latexrelease>
```

### 6.3 Undoing the temp modifications

If `\ExplSyntaxOn` exists (defined and not equal `\relax`), then use the `expl3` restore code, otherwise restore `\ExplSyntaxOn` and `\ExplSyntaxOff` to be undefined.

```
137 \*latexrelease-finish)
138 \ifundefined{ExplSyntaxOn}%
139   {\let\ExplSyntaxOn\@undefined
140    \let\ExplSyntaxOff\@undefined
141    \@gobble}%
142   {\ExplSyntaxOn
143    \@firstofone}%
144   {%
```

Now just loop through the list of redefined commands and restore their previous meanings.

```
145 \tl_map_inline:Nn \l_@@_restores_tl
146 {
147   \cs_set_eq:Nc #1 { @@_ \cs_to_str:N #1 }
148   \cs_undefine:c { @@_ \cs_to_str:N #1 }
149 }
150 \tl_clear:N \l_@@_restores_tl
```

And restore the silenced error messages.

```
151 \msg_redirect_name:nnn { hooks } { exists } { }
```

```

152 \msg_redirect_name:nnn { mark } { class-already-defined } { }
153 \msg_redirect_name:nnn { socket } { already-declared } { }
154 \msg_redirect_name:nnn { socket } { plug-already-declared } { }
155 <@@=>
156 \ExplSyntaxOff}%
157 </latexrelease-finish>

```

## 6.4 Individual Changes

The code for each change will be inserted at this point, extracted from the kernel source files.

## 6.5 fixltx2e

Generate a stub fixltx2e package:

```

158 (*fixltx2e)
159 \IncludeInRelease{2015/01/01}{\fixltxe}{Old fixltx2e package}
160 \NeedsTeXFormat{LaTeX2e}
161 \PackageWarningNoLine{fixltx2e}{%
162 fixltx2e is not required with releases after 2015\MessageBreak
163 All fixes are now in the LaTeX kernel.\MessageBreak
164 See the latexrelease package for details}
165 \EndIncludeInRelease
166 \IncludeInRelease{0000/00/00}{\fixltxe}{Old fixltx2e package}
167 \def\@outputdblcol{%
168   \if@firstcolumn
169     \global\@firstcolumnfalse
170     \global\setbox\@leftcolumn\copy\@outputbox
171     \splitmaxdepth\maxdimen
172     \vbadness\maxdimen
173     \setbox\@outputbox\vbox{\unvbox\@outputbox\unskip}%
174     \setbox\@outputbox\vsplit\@outputbox to\maxdimen
175     \toks@\expandafter{\topmark}%
176     \xdef\@firstcoltopmark{\the\toks@}%
177     \toks@\expandafter{\splitfirstmark}%
178     \xdef\@firstcolfirstmark{\the\toks@}%
179     \ifx\@firstcolfirstmark\@empty
180       \global\let\@setmarks\relax
181     \else
182       \gdef\@setmarks{%
183         \let\firstmark\@firstcolfirstmark
184         \let\topmark\@firstcoltopmark}%
185     \fi
186   \else
187     \global\@firstcolumntrue
188     \setbox\@outputbox\vbox{%
189       \hb@xt@\textwidth{%
190         \hb@xt@\columnwidth{\box\@leftcolumn \hss}%
191         \hfil
192         {\normalcolor\vrule \@width\columnseprule}%
193         \hfil
194         \hb@xt@\columnwidth{\box\@outputbox \hss}}}%
195   \@combinedblfloats

```

```

196 \setmarks
197 \outputpage
198 \begingroup
199 \dblfloatplacement
200 \startdblcolumn
201 \@whilesw\if@fcolmade \fi{\@outputpage\startdblcolumn}%
202 \endgroup
203 \fi}
204 \def\end@dblfloat{%
205 \if@twocolumn
206 \endfloatbox
207 \ifnum\@floatpenalty <\z@
208 \@largefloatcheck
209 \global\dp\@currbox1sp %
210 \cons\@currlist\@currbox
211 \ifnum\@floatpenalty <-\@Mii
212 \penalty -\@Miv
213 \@tempdima\prevdepth
214 \vbox{}}%
215 \prevdepth\@tempdima
216 \penalty\@floatpenalty
217 \else
218 \adjust{\penalty -\@Miv \vbox{}}\penalty\@floatpenalty}\@Esphack
219 \fi
220 \fi
221 \else
222 \end@float
223 \fi
224 }
225 \def\@testwrongwidth #1{%
226 \ifdim\dp#1=f@depth
227 \else
228 \global\@testtrue
229 \fi}
230 \let f@depth\z@
231 \def\@dblfloatplacement{\global\@dbltopnum\c@dbltopnumber
232 \global\@dbltoproom \dbltopfraction\@colht
233 \textmin \colht
234 \advance \textmin -\@dbltoproom
235 \fpmmin \dblfloatpagefraction\textheight
236 \fptop \dblfpfptop
237 \fpsep \dblfpsep
238 \fpbot \dblfpbot
239 \def f@depth{1sp}}
240 \def \doclearpage {%
241 \ifvoid\footins
242 \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
243 \setbox\@tempboxa\box\@cclv
244 \xdef\@deferlist{\@toplist\@botlist\@deferlist}%
245 \global \let \toplist \empty
246 \global \let \botlist \empty
247 \global \colroom \colht
248 \ifx \currlist\empty
249 \else

```

```

250         \@latexerr{Float(s) lost}\@ehb
251         \global \let \@currlist \@empty
252     \fi
253     \@makefcolumn\@deferlist
254     \@whiles\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
255     \if@twocolumn
256         \if@firstcolumn
257             \xdef\@deferlist{\@dbltoplist\@deferlist}%
258             \global \let \@dbltoplist \@empty
259             \global \@colht \textheight
260             \begingroup
261                 \dblfloatplacement
262                 \@makefcolumn\@deferlist
263                 \@whiles\if@fcolmade \fi{\@outputpage
264                     \@makefcolumn\@deferlist}%
265             \endgroup
266         \else
267             \vbox{}\clearpage
268         \fi
269     \fi
270     \ifx\@deferlist\@empty \else\clearpage \fi
271 \else
272     \setbox\@cclv\vbox{\box\@cclv\vfil}%
273     \@makecol\@opcol
274     \clearpage
275 \fi
276 }
277 \def \@startdblcolumn {%
278     \@tryfcolumn \@deferlist
279     \if@fcolmade
280     \else
281         \begingroup
282             \let \reserved@b \@deferlist
283             \global \let \@deferlist \@empty
284             \let \@elt \@sdblcolelt
285             \reserved@b
286         \endgroup
287     \fi
288 }
289 \def \@addtonextcol{%
290     \begingroup
291     \@insertfalse
292     \@setfloattypecounts
293     \ifnum \@fpstype=8
294     \else
295         \ifnum \@fpstype=24
296         \else
297             \@flsettextmin
298             \@reqcolroom \ht\@currbox
299             \advance \@reqcolroom \@textmin
300             \ifdim \@colroom>\@reqcolroom
301                 \@flsetnum \@colnum
302                 \ifnum\@colnum>\z@
303                     \@bitor\@currtype\@deferlist

```

```

304         \@testwrongwidth\@currbox
305         \if@test
306         \else
307             \@addtotopporbot
308         \fi
309     \fi
310 \fi
311 \fi
312 \fi
313 \if@insert
314 \else
315     \@cons\@deferlist\@currbox
316 \fi
317 \endgroup
318 }
319 \def\@addtodblcol{%
320 \begingroup
321 \insertfalse
322 \@setfloattypecounts
323 \@getfpsbit \tw@
324 \ifodd\@tempcnta
325     \@flsetnum \@dbltopnum
326     \ifnum \@dbltopnum>\z@
327         \@tempswafalse
328         \ifdim \@dbltoproom>\ht\@currbox
329             \@tempwattrue
330         \else
331             \ifnum \@fpstype<\sist@@n
332                 \advance \@dbltoproom \@textmin
333                 \ifdim \@dbltoproom>\ht\@currbox
334                     \@tempwattrue
335                 \fi
336                 \advance \@dbltoproom -\@textmin
337             \fi
338         \fi
339     \if@tempswa
340         \@bitor \@currtype \@deferlist
341         \@testwrongwidth\@currbox
342         \if@test
343         \else
344             \@tempdima -\ht\@currbox
345             \advance\@tempdima
346             -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
347                 \dblfloatsep \fi
348             \global \advance \@dbltoproom \@tempdima
349             \global \advance \@colht \@tempdima
350             \global \advance \@dbltopnum \m@ne
351             \@cons \@dbltoplist \@currbox
352             \inserttrue
353         \fi
354     \fi
355 \fi
356 \fi
357 \if@insert

```

```

358 \else
359 \cons\@deferlist\@currbox
360 \fi
361 \endgroup
362 }
363 \def \@addtocurcol {%
364 \insertfalse
365 \setfloatypecounts
366 \ifnum \@fpstype=8
367 \else
368 \ifnum \@fpstype=24
369 \else
370 \flsettextmin
371 \advance \@textmin \@textfloatsheight
372 \reqcolroom \pageht
373 \ifdim \@textmin>\reqcolroom
374 \reqcolroom \@textmin
375 \fi
376 \advance \reqcolroom \ht\@currbox
377 \ifdim \@colroom>\reqcolroom
378 \flsetnum \colnum
379 \ifnum \@colnum>\z@
380 \bitor\@currtype\@deferlist
381 \testwrongwidth\@currbox
382 \if@test
383 \else
384 \bitor\@currtype\@botlist
385 \if@test
386 \addtobot
387 \else
388 \ifodd \count\@currbox
389 \advance \reqcolroom \intextsep
390 \ifdim \@colroom>\reqcolroom
391 \global \advance \colnum \m@ne
392 \global \advance \@textfloatsheight \ht\@currbox
393 \global \advance \@textfloatsheight 2\intextsep
394 \cons \midlist \@currbox
395 \if@nobreak
396 \nobreak
397 \nobreakfalse
398 \everypar{}%
399 \else
400 \addpenalty \interlinepenalty
401 \fi
402 \vskip \intextsep
403 \box\@currbox
404 \penalty\interlinepenalty
405 \vskip\intextsep
406 \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi
407 \outputpenalty \z@
408 \inserttrue
409 \fi
410 \fi
411 \if@insert

```



```

412             \else
413             \@addtotoporbot
414             \fi
415         \fi
416     \fi
417 \fi
418 \fi
419 \fi
420 \fi
421 \if@insert
422 \else
423     \resetfps
424     \@cons\@deferlist\@currbox
425 \fi
426 }
427 \def\@xtryfc #1{%
428     \@next\reserved@a\@trylist{}\{}%
429     \@currtype \count #1%
430     \divide\@currtype\@xxxii
431     \multiply\@currtype\@xxxii
432     \@bitor \@currtype \@failedlist
433     \@testfp #1%
434     \@testwrongwidth #1%
435     \ifdim \ht #1>\@colht
436         \@testtrue
437     \fi
438     \if@test
439         \@cons\@failedlist #1%
440     \else
441         \@ytryfc #1%
442     \fi}
443 \def\@ztryfc #1{%
444     \@tempcnta\count #1%
445     \divide\@tempcnta\@xxxii
446     \multiply\@tempcnta\@xxxii
447     \@bitor \@tempcnta {\@failedlist \@flfail}%
448     \@testfp #1%
449     \@testwrongwidth #1%
450     \@tempdimb\@tempdima
451     \advance\@tempdimb\ht #1%
452     \advance\@tempdimb\@fpsep
453     \ifdim \@tempdimb >\@colht
454         \@testtrue
455     \fi
456     \if@test
457         \@cons\@flfail #1%
458     \else
459         \@cons\@flsucceed #1%
460         \@tempdima\@tempdimb
461     \fi}
462 \def\@{\spacefactor\@m{}}
463 \def\@tempa#1#2{#1#2\relax}
464 \ifx\setlength\@tempa
465     \def\setlength#1#2{#1 #2\relax}

```

```

466 \fi
467 \def\addpenalty#1{%
468   \ifvmode
469     \if@minipage
470     \else
471       \if@nobreak
472       \else
473         \ifdim\lastskip=\z@
474         \penalty#1\relax
475         \else
476         \@tempskipb\lastskip
477         \begingroup
478         \advance \@tempskipb
479           \ifdim\prevdepth>\maxdepth\maxdepth\else
480           \ifdim \prevdepth = -\@m\p@ \z@ \else \prevdepth \fi
481         \fi
482         \vskip -\@tempskipb
483         \penalty#1%
484         \vskip \@tempskipb
485       \endgroup
486       \vskip -\@tempskipb
487       \vskip \@tempskipb
488     \fi
489   \fi
490 \fi
491 \else
492   \@noitemerr
493 \fi}
494 \def\@fnsymbol#1{%
495   \ifcase#1\or \TextOrMath\textasteriskcentered *\or
496   \TextOrMath \textdagger \dagger\or
497   \TextOrMath \textdaggerdbl \ddagger \or
498   \TextOrMath \textsection \mathsection\or
499   \TextOrMath \textparagraph \mathparagraph\or
500   \TextOrMath \textbardbl ||\or
501   \TextOrMath {\textasteriskcentered\textasteriskcentered}{**}\or
502   \TextOrMath {\textdagger\textdagger}{\dagger\dagger}\or
503   \TextOrMath {\textdaggerdbl\textdaggerdbl}{\ddagger\ddagger}\else
504   \@ctrerr \fi
505 }
506 \begingroup\expandafter\expandafter\expandafter\endgroup
507 \expandafter\ifx\csname eTeXversion\endcsname\relax
508 \DeclareRobustCommand\TextOrMath{%
509   \ifmmode \expandafter\@secondoftwo
510   \else \expandafter\@firstoftwo \fi}
511 \protected@edef\TextOrMath#1#2{\TextOrMath{#1}{#2}}
512 \else
513 \protected\expandafter\def\csname TextOrMath\space\endcsname{%
514   \ifmmode \expandafter\@secondoftwo
515   \else \expandafter\@firstoftwo \fi}
516 \edef\TextOrMath#1#2{%
517   \expandafter\noexpand\csname TextOrMath\space\endcsname
518   {#1}{#2}}
519 \fi

```

```

520 \def\@esphack{%
521   \relax
522   \ifhmode
523     \spacefactor\@savsf
524     \ifdim\@savsk>\z@
525       \nobreak \hskip\z@skip % <-----
526       \ignorespaces
527     \fi
528   \fi}
529 \def\@Esphack{%
530   \relax
531   \ifhmode
532     \spacefactor\@savsf
533     \ifdim\@savsk>\z@
534       \nobreak \hskip\z@skip % <-----
535       \@ignoretrue
536       \ignorespaces
537     \fi
538   \fi}
539 \DeclareRobustCommand\em
540   {\@nomath\em \ifdim \fontdimen\@one\font >\z@
541     \eminnershape \else \itshape \fi}
542 \def\eminnershape{\upshape}
543 \DeclareRobustCommand*\textsubscript[1]{%
544   \@textsubscript{\selectfont#1}}
545 \def\@textsubscript#1{%
546   {\m@th\ensuremath_{\mbox{\fontsize\sf@size\z@#1}}}}
547 \def\@DeclareMathSizes #1#2#3#4#5{%
548   \@defaultunits\dimen@ #2pt\relax\@nnil
549   \if $#3$%
550     \expandafter\let\csname S@\strip@pt\dimen@\endcsname\math@fontsfalse
551   \else
552     \@defaultunits\dimen@ii #3pt\relax\@nnil
553     \@defaultunits\@tempdima #4pt\relax\@nnil
554     \@defaultunits\@tempdimb #5pt\relax\@nnil
555     \toks@{#1}%
556     \expandafter\xdef\csname S@\strip@pt\dimen@\endcsname{%
557       \gdef\noexpand\tf@size{\strip@pt\dimen@ii}%
558       \gdef\noexpand\sf@size{\strip@pt\@tempdima}%
559       \gdef\noexpand\ssf@size{\strip@pt\@tempdimb}%
560       \the\toks@
561     }%
562   \fi
563 }
564 \providecommand*\MakeRobust[1]{%
565   \@ifundefined{\expandafter\@gobble\string#1}{%
566     \@latex@error{The control sequence '\string#1' is undefined!%
567     \MessageBreak There is nothing here to make robust}%
568     \@eha
569   }%
570   {%
571     \@ifundefined{\expandafter\@gobble\string#1\space}%
572     {%
573       \expandafter\let\csname

```

```

574 \expandafter\@gobble\string#1\space\endcsname=#1%
575 \edef\reserved@a{\string#1}%
576 \def\reserved@b{#1}%
577 \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
578 \edef#1{%
579 \ifx\reserved@a\reserved@b
580 \noexpand\x@protect\noexpand#1%
581 \fi
582 \noexpand\protect\expandafter\noexpand
583 \csname\expandafter\@gobble\string#1\space\endcsname}%
584 }%
585 {\@latex@info{The control sequence '\string#1' is already robust}}%
586 }%
587 }
588 \MakeRobust\ (
589 \MakeRobust\ )
590 \MakeRobust\ [
591 \MakeRobust\ ]
592 \MakeRobust\ makebox
593 \MakeRobust\ savebox
594 \MakeRobust\ framebox
595 \MakeRobust\ parbox
596 \MakeRobust\ rule
597 \MakeRobust\ raisebox
598 \def\@xfloat #1[#2]{%
599 \@nodocument
600 \def \@cuptype {#1}%
601 \def \@fps {#2}%
602 \@onelevel@sanitize \@fps
603 \def \reserved@b {!}%
604 \ifx \reserved@b \@fps
605 \@fpsadddefault
606 \else
607 \ifx \@fps \@empty
608 \@fpsadddefault
609 \fi
610 \fi
611 \ifhmode
612 \@bsphack
613 \@floatpenalty -\@Mii
614 \else
615 \@floatpenalty -\@Miii
616 \fi
617 \ifinner
618 \@parmoderr\@floatpenalty\z@
619 \else
620 \@next\@currbox\@freelist
621 {%
622 \@tempcnta \sixt@@n
623 \expandafter \@tfor \expandafter \reserved@a
624 \expandafter :\expandafter =\@fps
625 \do
626 {%
627 \if \reserved@a h%

```

```

628         \ifodd \@tempcnta
629         \else
630             \advance \@tempcnta \@ne
631         \fi
632     \else\if \reserved@a t%
633         \@setfpsbit \tw@
634     \else\if \reserved@a b%
635         \@setfpsbit 4%
636     \else\if \reserved@a p%
637         \@setfpsbit 8%
638     \else\if \reserved@a !%
639         \ifnum \@tempcnta>15
640             \advance\@tempcnta -\sist@n\relax
641         \fi
642     \else
643         \@latex@error{Unknown float option '\reserved@a'}%
644         {Option '\reserved@a' ignored and 'p' used.}%
645         \@setfpsbit 8%
646     \fi\fi\fi\fi\fi
647 }%
648 \@tempcntb \csname ftype@\@capytype \endcsname
649 \multiply \@tempcntb \@xxxii
650 \advance \@tempcnta \@tempcntb
651 \global \count\@currbox \@tempcnta
652 }%
653 \f@floatovf
654 \fi
655 \global \setbox\@currbox
656 \color@vbox
657 \normalcolor
658 \vbox \bgroup
659 \hsize\columnwidth
660 \@parboxrestore
661 \@floatboxreset
662 }
663 \def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}
664 \EndIncludeInRelease
665 \fixltx2e

```