The **latexrelease** package*

The LaTeX Project

2024/08/11

> This file is maintained by the LaTeX Project team.
> Bug reports can be opened (category `latex`) at
> https://latex-project.org/bugs.html.

1 Introduction

Prior to the 2015 release of LaTeX, essentially no changes had been made to the LaTeX 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 LaTeX 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 LaTeX, 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 LaTeX 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 LaTeX that is in force, so affecting all other packages used in the document.

---

*This file has version number v1.0p, last revised 2024/08/11.

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 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 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 LaTeX release date as shown at the start of every 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 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 LaTeX release or from CTAN and distributed with the document, it will make an older 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 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 LaTeX, 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 LaTeX 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}
  ...
```

4

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.[1]

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 LaTeX 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*⟩.

---

[1]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 LaTeX kernel, taking lthooks as example:

```
%<*2ekernel|latexrelease>
\ExplSyntaxOn
%<latexrelease>\NewModuleRelease{2020/10/01}{lthooks}%
%<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}{lthooks}{Undefine~lthooks}
%<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 LaTeX release updates to the LaTeX 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 LaTeX 2_ε. Earlier ones where really quite different from one another.

```
1 ⟨∗latexrelease⟩
2 \NeedsTeXFormat{LaTeX2e}[1996/06/01]
```

6.1   Setup

\sourceLaTeXdate   Store the original LaTeX 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 LaTeX 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]{\@IncludeInRele@se{#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@postltexpl{\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 { \renew@command } { \new@command } #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:nnx` 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:nnx \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 wor
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 }
```

Now a one-off for `ltcmd.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@postltexpl` actions above:

```
125 \cs_gset_protected:Npn \latexrelease@postltcmd
126   {
127     \@@_redefines:w
128       \NewDocumentCommand \DeclareDocumentCommand
129       \q_recursion_tail \q_recursion_tail
130       \q_recursion_stop
131   }
132   }}%
133 \endgroup
134 ⟨/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.

```
135 ⟨∗latexrelease-finish⟩
136 \@ifundefined{ExplSyntaxOn}%
137   {\let\ExplSyntaxOn\@undefined
138    \let\ExplSyntaxOff\@undefined
139    \@gobble}%
140   {\ExplSyntaxOn
141    \@firstofone}%
142   {%
```

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

```
143 \tl_map_inline:Nn \l_@@_restores_tl
144   {
145     \cs_set_eq:Nc #1 { @@_ \cs_to_str:N #1 }
146     \cs_undefine:c { @@_ \cs_to_str:N #1 }
147   }
148 \tl_clear:N \l_@@_restores_tl
```

And restore the silenced error messages.

```
149 \msg_redirect_name:nnn { hooks } { exists } { }
150 \msg_redirect_name:nnn { mark } { class-already-defined } { }
151 ⟨@@=⟩
152   \ExplSyntaxOff}%
153 ⟨/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:

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

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

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

```
308    \fi
309    \if@insert
310    \else
311      \@cons\@deferlist\@currbox
312    \fi
313   \endgroup
314 }
315 \def\@addtodblcol{%
316   \begingroup
317    \@insertfalse
318    \@setfloattypecounts
319    \@getfpsbit \tw@
320    \ifodd\@tempcnta
321     \@flsetnum \@dbltopnum
322     \ifnum \@dbltopnum>\z@
323       \@tempswafalse
324       \ifdim \@dbltoproom>\ht\@currbox
325         \@tempswatrue
326       \else
327        \ifnum \@fpstype<\sixt@@n
328          \advance \@dbltoproom \@textmin
329          \ifdim \@dbltoproom>\ht\@currbox
330            \@tempswatrue
331          \fi
332          \advance \@dbltoproom -\@textmin
333        \fi
334       \fi
335       \if@tempswa
336          \@bitor \@currtype \@deferlist
337         \@testwrongwidth\@currbox
338          \if@test
339          \else
340            \@tempdima -\ht\@currbox
341            \advance\@tempdima
342              -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
343                                        \dblfloatsep \fi
344            \global \advance \@dbltoproom \@tempdima
345            \global \advance \@colht \@tempdima
346            \global \advance \@dbltopnum \m@ne
347            \@cons \@dbltoplist \@currbox
348              \@inserttrue
349          \fi
350        \fi
351     \fi
352    \fi
353    \if@insert
354    \else
355      \@cons\@deferlist\@currbox
356    \fi
357   \endgroup
358 }
359 \def \@addtocurcol {%
360    \@insertfalse
361    \@setfloattypecounts
```

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

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

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

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

```
578        \noexpand\protect\expandafter\noexpand
579        \csname\expandafter\@gobble\string#1\space\endcsname}%
580     }%
581     {\@latex@info{The control sequence '\string#1' is already robust}}%
582     }%
583 }
584 \MakeRobust\(
585 \MakeRobust\)
586 \MakeRobust\[
587 \MakeRobust\]
588 \MakeRobust\makebox
589 \MakeRobust\savebox
590 \MakeRobust\framebox
591 \MakeRobust\parbox
592 \MakeRobust\rule
593 \MakeRobust\raisebox
594 \def\@xfloat #1[#2]{%
595   \@nodocument
596   \def \@captype {#1}%
597   \def \@fps {#2}%
598   \@onelevel@sanitize \@fps
599   \def \reserved@b {!}%
600   \ifx \reserved@b \@fps
601     \@fpsadddefault
602   \else
603     \ifx \@fps \@empty
604       \@fpsadddefault
605     \fi
606   \fi
607   \ifhmode
608     \@bsphack
609     \@floatpenalty -\@Mii
610   \else
611     \@floatpenalty-\@Miii
612   \fi
613   \ifinner
614     \@parmoderr\@floatpenalty\z@
615   \else
616     \@next\@currbox\@freelist
617       {%
618        \@tempcnta \sixt@@n
619        \expandafter \@tfor \expandafter \reserved@a
620          \expandafter :\expandafter =\@fps
621          \do
622            {%
623             \if \reserved@a h%
624               \ifodd \@tempcnta
625               \else
626                 \advance \@tempcnta \@ne
627               \fi
628             \else\if \reserved@a t%
629               \@setfpsbit \tw@
630             \else\if \reserved@a b%
631               \@setfpsbit 4%
```

```
632        \else\if \reserved@a p%
633          \@setfpsbit 8%
634        \else\if \reserved@a !%
635          \ifnum \@tempcnta>15
636            \advance\@tempcnta -\sixt@@n\relax
637          \fi
638        \else
639          \@latex@error{Unknown float option `\reserved@a'}%
640          {Option `\reserved@a' ignored and `p' used.}%
641          \@setfpsbit 8%
642        \fi\fi\fi\fi\fi
643        }%
644      \@tempcntb \csname ftype@\@captype \endcsname
645      \multiply \@tempcntb \@xxxii
646      \advance \@tempcnta \@tempcntb
647      \global \count\@currbox \@tempcnta
648      }%
649    \@fltovf
650  \fi
651  \global \setbox\@currbox
652    \color@vbox
653      \normalcolor
654      \vbox \bgroup
655        \hsize\columnwidth
656        \@parboxrestore
657        \@floatboxreset
658 }
659 \def\@stpelt#1{\global\csname c@#1\endcsname \m@ne\stepcounter{#1}}
660 \EndIncludeInRelease
661 ⟨/fixltx2e⟩
```