xnotes2bib — Integrating notes into the bibliography*

Joseph Wright†

Released 2008/10/22

Abstract

The xnotes2bib package is an *experimental* re-implementation of notes2bib using LaTeX3 commands internally (taken from the expl3 bundle). The user interface is almost identical to notes2bib.

The xnotes2bibpackage defines a new type of note, `\bibnote`, which will always be added to the bibliography. The package allows footnotes and endnotes to be moved into the bibliography in the same way. The package can be used with natbib and biblatex as well as plain LaTeX citations. Both sorted and unsorted bibliography styles are supported.

Contents

*This file describes version v5, last revised 2008/10/22.

†E-mail: joseph.wright@morningstar2.co.uk

1 Introduction

The new code syntax introduced by LaTeX3 is very different from plain TeX or LaTeX 2$_\varepsilon$. This means that converting packages to LaTeX3 requires working from the ground up if the new ideas are to be exploited fully. This package re-implements notes2bib using the experimental syntax for internal commands.

The internal logic of the package remains broadly similar to that in notes2bib. However, some opportunities to simplify the code or make the flow clearer have been taken. There is therefore no one-to-one correspondence between the internals of the two packages. In particular, this package does not guarantee backward-compatibility with notes2bib under all circumstances. LaTeX3 will require significant changes to documents, and so alterations to the logic of packages in order to make them more useful in the long term seem sensible.

2 Using the package

2.1 Macros

`\bibnote`
`\bibnotemark`
`\bibnotetext`
`\flushnotestack`

The user macros of xnotes2bib are intended to work in the same way as those in notes2bib. Please see `notes2bib.pdf` for details of:

- `\bibnote`
- `\bibnotemark`
- `\bibnotetext`
- `\flushnotestack`

The `\citenote` macro can be replaced by using `\bibnotemark` to cite the note for a second time.

2.2 Control values

`\niibsetup`

The main user changes between notes2bib and xnotes2bib are in the user options. These have been completely re-written here, to improve the logic of the interface and to remove unnecessary options. All package control is currently carried out using the `\niibsetup` macro, as the LaTeX 2$_\varepsilon$ kernel expands options and removes spaces. As in notes2bib, the `\niibsetup` macro accepts a key–value list as input.

There are four types of control key. Boolean keys take the values `true` and `false`. If the key name alone is given, this is the same as giving the `true` value. String keys save their value as a string for internal use: they can therefore take any input. Choice keys take

one of a range of pre-determined values, in a similar way to Boolean options. However, the list of possible values varies from key to key. Finally, macro keys require a macro as the value.

`convert endnotes`
`convert footnotes`

To convert endnotes and footnotes into bibnotes, the options `convert endnotes` and `convert footnotes` are available. Both taken Boolean values, and are set to `false` when loading xnotes2bib. If endnotes are not defined elsewhere, the `convert endnotes` only defines them when set to `true`.

`placement`

The package is able to position notes in three ways in the bibliography. The standard method is to create the notes interspersed with "real" citations. The order in the source then determines the position in the bibliography. The package can also create notes such that they appear either before or after normal citations. The behaviour here is controlled by the `placement` option, which takes a choice from the list `after`, `before` and `mixed`. The `after` option places notes after normal citations, `before` places notes first and `mixed` mixes notes and citations based on the order in the source.

`key after`
`key before`
`key mixed`

Most sorted bibliography styles provide methods for controlling the sort order of entries. With standard BibTeX styles, the `key` field is used for this purpose. The options `key after`, `key before` and `key mixed` are used to set this up. They contain an item to be added before the note label when creating the `key` field. The standard values are `aaa` for `key before` and `zzz` for `key after`: `key mixed` is empty. When using biblatex, `presort` field is available. The values used by xnotes2bib for this field are stored as `presort after`, `presort before` and `presort mixed`, with default values `mn`, `ml` and `mm`, respectively. The `presort` system is described in the biblatex manual.

`presort after`
`presort before`
`presort mixed`

`write key field`

When using the package with natbib, it may be necessary to avoid writing the `key` field to the temporary database. This is seen with the style `unsrtnat`, for example. The Boolean option `write key field` is available to turn off writing the `key` field under these circumstances.

`cite`

As xnotes2bib uses BibTeX to create notes, it uses a citation command to mark the notes in the body of the text. The command used is controlled by the macro key `cite`. The standard value is `\cite`.

`field name prefix record`

Four control values are available to affect low-level aspects of the creation of the notes database. The `prefix` option sets the string used by xnotes2bib to name the database, and is set to `niib-` initially. The `field` and `record` options contain the name of the field used for the note, and the BibTeX record type used. The standard values are `note` and `Misc`, respectively. Finally, the `name` option sets the text used in addition to a number when automatically labelling notes. This is set to `Bibnote`: there is no space between this text and the number as this will confuse BibTeX.

`debug`

The `debug` Boolean option is available for understanding what the package is doing. When set to `true`, the package writes additional information to the log to aid problem solving.

3 Testing the package

This release of xnotes2bib has several aims, all related to testing. Firstly, it provides a method to examine new ideas for notes2bib: feedback on the altered options is welcome. Second, it is a test-bed for using expl3 to create packages. People familiar with the current LaTeX3 ideas are encouraged to read the source and make suggestions. The new coding environment requires a more structured approach than LaTeX 2ε, and so it is possible ideas have been missed.

The author wonders if this package could be implemented using the template system and xfootnotes. Suggestions in this area are welcome!

In order to use xnotes2bib, you will need an up-to-date TeX system and recent copies of expl3 and xparse. The package is tested with the current SVN release of the LaTeX3 packages, but these may change on a day-to-day basis. So testers should be prepared for problems such as undefined macros from the new kernel! The ε-TeX extensions to TeX are required by xnotes2bib.

Due to the experimental nature of the package, it currently has no version number. The package date should be used to check if versions are identical. That said, the version of notes2bib will be stepped when new copies of xnotes2bib are created.

While the package is experimental, it should work properly as a replacement for notes2bib, with the proviso that the options are very different. Any problems of function should be reported as normal.

4 Installation

The entire bundle is supplied with the TDS-ready `.zip` file `notes2bib.tds.zip`. Simply unzip this into your local texmf tree and run your hash program (`texhash` for TeXLive or `initextmf -u` for MiKTeX)

To extract the package `xnotes2bib.sty` from `xnotes2bib.dtx`, run (pdf)LaTeX on the later. This will also create the documentation and `README.txt` file. Three (pdf)LaTeX runs with `\write18` enabled will also build the index and table of contents in the `.pdf`.

5 Implementation

5.1 Preliminaries

The package starts by loading the basics of expl3. To keep things clear, first l3names is loaded to do the basic identification, and then everything else is started up. This is based on Morten Høgholm's xfrac package lead-off.

```
⟨*package⟩
\ProvidesExplPackage
  {\filename}{\filedate}{\fileversion}{\filedescription}
\RequirePackage {expl3,keys3,xparse}
```

As the expl3 syntax is in use, there is no need to worry about white space in the code. So there are very few % characters in the following. As far as possible, expl3 or xparse macros are used here; however, a lot of the user-space LaTeX 2ε macros should still be valid in LaTeX3. There are also a few @ macros where things are tidied closely to the LaTeX 2ε kernel at present (for example \@filesw).

\c_niib_err_tlp
\l_niib_packge_option_tlp

Error messages are created.

```
\err_file_new:Nn \c_niib_err_tlp {xnotes2bib.err}
\tlp_new:Nn \l_niib_err_option_tlp {option}
\err_interrupt_new:NNNnnn \c_niib_err_tlp \l_niib_err_option_tlp 0
  {All~package~control~should~be~given~using~the~\token_to_str:N~
   \niibsetup~\text_put_sp:~macro}
  {\err_help_return_or_X:}
  {}
\err_file_close:N \c_niib_err_tlp
```

As the LaTeX 2ε kernel expands package options, for the moment xnotes2bib does not accept any load-time options.

```
\DeclareOption* {
  \err_interrupt:NNw \c_niib_err_tlp \l_niib_err_option_tlp
}
\ProcessOptions \scan_stop:
```

\g_niib_note_int

A counter for the automatically-created notes is needed. This is a global value (life will get very complicated if not).

```
\int_new:N \g_niib_note_int
```

\niib_thenote:

The counter needs to be expressed as a usable name. A check is made so that if there are more than ten notes then the number is padded.

```
\def_new:Npn \niib_thenote: {
  \l_niib_name_tlp
  \num_compare:nNnT {\g_niib_totalnotes_int} > {\c_nine} {
    \num_compare:nNnF {\g_niib_note_int} > {\c_nine} {
      0
    }
  }
  \int_to_arabic:n \g_niib_note_int
}
```

5.2 User options

\l_niib_after_cite_bool
\l_niib_before_cite_bool

The placement option needs two Boolean values.

```
\bool_new:N \l_niib_after_cite_bool
\bool_new:N \l_niib_before_cite_bool
```

\niib_convert_endnotes:
\niib_convert_footnotes:

Some set up for other options.

```
\let_new:NN \niib_convert_endnotes: \scan_stop:
\let_new:NN \niib_convert_footnotes: \scan_stop:
```

\niib_cite:n

The cite key is stored.

```
\keys_manage_quick:n {
  niib/.cd:,
  cite/.set:N                    = \niib_cite:n,
```

\niib_debug:n

The debug option is implemented as a choice.

```
  debug/.choice:,
  debug/.default:n               = true,
  debug/true/.code:n             = {
    \def:Npn \niib_debug:n ##1 {
      \err_info:nn {##1} {(xnotes2bib) \text_put_four_sp:}
    }
  },
  debug/false/.code:n            = {
    \let:NN \niib_debug:n \use_none:n
  },
```

The convert footnotes and convert endnotes options work as choices.

```
  convert~endnotes/.choices:nn = {true,false} {
    \let:Nc \niib_convert_endnotes:
      {niib_convert_endnotes_ \l_keys_current_choice_tlp}
    },
  convert~endnotes/.default:n     = true,
  convert~footnotes/.choices:nn = {true,false} {
    \let:Nc \niib_convert_footnotes:
      {niib_convert_footnotes_ \l_keys_current_choice_tlp}
    },
  convert~footnotes/.default:n     = true,
```

\l_niib_field_tlp
\l_niib_key_after_tlp
\l_niib_key_before_tlp
\l_niib_key_mixed_tlp
\l_niib_keyword_tlp
\l_niib_name_tlp

Some storage macros.

```
  /keys/current~module:n          = niib,
```

```
  field/.set:c                    = field,
  key~after/.set:c                = key_after,
  key~before/.set:c               = key_before,
  key~mixed/.set:c                = key_mixed,
  keyword/.set:c                  = keyword,
  name/.set:c                     = name,
```

Another choice for placement.

```
  placement/.choice:              = \l_niib_placement_tlp,
  placement/.default:n            = mixed,
  placement/after/.code:n         = {
    \bool_set_true:N  \l_niib_after_cite_bool
    \bool_set_false:N \l_niib_before_cite_bool
  },
  placement/before/.code:n        = {
    \bool_set_false:N \l_niib_after_cite_bool
    \bool_set_true:N \l_niib_before_cite_bool
  },
  placement/mixed/.code:n         = {
    \bool_set_false:N \l_niib_after_cite_bool
    \bool_set_false:N \l_niib_before_cite_bool
  },
```

\l_niib_prefix_tlp
\l_niib_presort_after_tlp
\l_niib_presort_before_tlp
\l_niib_presort_mixed_tlp
\l_niib_record_tlp

```
  /keys/current~module:n          = niib,
  prefix/.set:c                   = prefix,
  presort~after/.set:c            = presort_after,
  presort~before/.set:c           = presort_before,
  presort~mixed/.set:c            = presort_mixed,
  record/.set:c                   = record,
```

\l_niib_write_key_bool

One real Boolean key.

```
  write~key~field/.boolean:c      = write_key
}
```

All of the default values are set.

```
\keys_manage_quick:n{
  /niib/.cd:,
  cite            = \cite,
  debug           = false,
  field           = note,
  key~after       = zzz,
  key~before      = aaa,
  key~mixed       = {},
  name            = Bibnote,
```

```
  placement       = mixed,
  prefix          = niib-,
  presort~after   = mn,
  presort~before  = ml,
  presort~mixed   = mm,
  record          = Misc,
  write~key~field = true}
```

5.3 User macros

`\bibnote [#1]:` label

`#2:` text

The main user macro has to check if the automatic note number is needed. If it is, the optional argument is note used to `\bibnotetext` as this automatically assumes the current value of `\niib_thenote:`. The note text must be sorted out before the mark is made, as otherwise packages such as cite do not work correctly with punctuation. As the package is designed for LaTeX3, it is assumed ε-TeX is available and so `\exp_not:n` can be used on the mandatory argument. Hence everything is absorbed here without worrying about category codes.

```
\NewDocumentCommand {\bibnote} {o>{P}m} {
  \IfNoValueTF {#1} {
    \int_gincr:N \g_niib_note_int
    \bibnotetext {#2}
    \bibnotemark [\niib_thenote:]
  }{
    \bibnotetext [#1] {#2}
    \bibnotemark [#1]
  }
}
```

`\bibnotemark [#1]:` label

As with `\bibnote`, if no optional argument is given the global note tracking counter is incremented before carrying out the citation.

```
\NewDocumentCommand {\bibnotemark} {o} {
  \IfNoValueTF {#1} {
    \int_gincr:N \g_niib_note_int
    \niib_notemark:n {\niib_thenote:}
  }{
    \niib_notemark:n {#1}
  }
}
```

`\bibnotetext [#1]:` label

`#2:` text

Writing the note text to the database should only occur if files are being written. For the moment, this uses the LaTeX 2_ε flag (and so the tradition syntax).

```
\NewDocumentCommand {\bibnotetext} {O{\niib_thenote:}>{P}m} {
  \if@filesw
```

`\g_niib_out_stream` A new output stream may need to be created, if this is the first note to write. The LaTeX 2_ε file switch is checked to see if this should occur.

```
    \cs_if_free:NT \g_niib_out_stream {
      \iow_new:N \g_niib_out_stream
      \iow_open:Nn \g_niib_out_stream
        {\l_niib_prefix_tlp \c_job_name_tlp.bib}
      \iow_expanded:Nn \g_niib_out_stream
        {This~is~an~auxiliary~file~used~by~the~‘xnotes2bib’~package.^^J
         This~file~may~safely~be~deleted.^^J It~will~be~recreated~as~
         required.^^J}
    }
```

The writing macro works in expanded mode, as everything except the note text does need expansion. Notice that `\space` has to be used here after the macro names, as `~` does not work.

```
    \niib_debug:n {
      Writing~text:\MessageBreak #2\MessageBreak to~database~for~note~
      ‘#1’
    }
    \iow_expanded:Nn \g_niib_out_stream {
      @\l_niib_record_tlp{
        #1,^^J
        \text_put_sp: \text_put_sp: \l_niib_field_tlp \text_put_sp:
          =~{\exp_not:n{#2}},^^J
        \bool_if:NT \l_niib_write_key_bool {
          \text_put_sp: \text_put_sp: \l_niib_keyname_tlp \text_put_sp:
            =~{\niib_key:#1},^^J
        }
        \text_put_sp: \text_put_sp: keywords~
          =~{\l_niib_keyword_tlp},^^J
        \text_put_sp: \text_put_sp: presort~
          =~{\niib_presort:},^^J
      }^^J
    }
  \fi
}
```

`\flushnotestack` In order to delay citations to the end of the bibliography (and thus force others to the start), note citations are held in a clist until being added to the `.aux` file using `\flushnotestack`. At that point, the stack is cleared so that collection can begin again.

```
\NewDocumentCommand {\flushnotestack} {} {
  \clist_if_empty:NF \g_niib_before_cite_clist {
    \if@filesw
      \iow_expanded:Nn \@auxout {
        \token_to_str:N \niibbeforecite {
          \g_niib_before_cite_clist
        }
      }
    \fi
  }
  \clist_if_empty:NF \g_niib_after_cite_clist {
    \nocite { \g_niib_after_cite_clist}
    \if@filesw
      \iow_expanded:Nn \@auxout {
        \token_to_str:N \niibaftercite {\g_niib_after_cite_clist}
      }
    \fi
  }
  \clist_gclear:N \g_niib_after_cite_clist
}
```

`\niibsetup` The document-level command for settings.

```
\NewDocumentCommand {\niibsetup} {m} {\keys_manage:n {/niib/.cd:,#1}}
```

5.4 Macros for the `.aux` file

To allow tracking of the need to re-run LaTeX, each run writes the current out-of-order citations to the `.aux` file. This requires two user-space macros, although it is not intended the user ever applies them!

`\g_niib_before_cite_old_clist` `\g_niib_after_cite_old_clist` Two lists are created to track the out-of-order citations written in the previous run to the `.aux` file.

```
\clist_new:N \g_niib_before_cite_old_clist
\clist_new:N \g_niib_after_cite_old_clist
```

`\g_niib_document_tlp` A hook into the `\document` macro is needed by `\niibbeforecite`; this needs to exist before `\document` is executed, hence needing a tlp for this.

```
\tlp_new:N \g_niib_document_tlp
\tlp_gput_right:Nn \document {\g_niib_document_tlp}
```

`\niibbeforecite` `\niibaftercite` `#1:` citations

The macros themselves are quite simple. `\clist_gset_eq:Nc` is not used as the macros could appear more than once in the `.aux` file.

```
\NewDocumentCommand {\niibbeforecite} {m} {
  \clist_gput_right:Nn \g_niib_before_cite_old_clist {#1}
  \tlp_gput_right:Nn \g_niib_document_tlp {\nocite{#1}}
}
\NewDocumentCommand {\niibaftercite} {m} {
  \clist_gput_right:Nn \g_niib_after_cite_old_clist {#1}
}
```

`\g_niib_totalnotes_int` The total number of notes in the last run is needed

```
\int_new:N \g_niib_totalnotes_int
```

`\niibtotalnotes` `#1:` number

The total number of notes created is stored for the next run.

```
\NewDocumentCommand {\niibtotalnotes} {m} {
  \int_gset:Nn \g_niib_totalnotes_int {#1}
}
```

5.5 Internal macros

`\niib_notemark:n` The citation command itself is in an internal macro, as it is called from a few places. Both the `before` and `after` modes require some tracking of the citations.

```
\def_new:Npn \niib_notemark:n #1 {
  \niib_debug:n {Creating~citation~for~note~‘#1’}
  \bool_if:NT \l_niib_before_cite_bool {
    \niib_debug:n {Note~‘#1’~before~real~citations}
    \clist_gput_right:Nx \g_niib_before_cite_clist {#1}
  }
  \bool_if:NTF \l_niib_after_cite_bool {
    \niib_debug:n {Note~‘#1’~after~real~citations}
    \clist_gput_right:Nx \g_niib_after_cite_clist {#1}
    \niib_after_cite:n {#1}
  }{
    \niib_cite:n {#1}
  }
}
```

`\g_niib_before_cite_clist` `\g_niib_after_cite_clist` Out-of-order citations in the current run are followed using two clists.

```
\clist_new:N \g_niib_before_cite_clist
\clist_new:N \g_niib_after_cite_clist
```

`\niib_key:` `\niib_presort:` The key and presort information for the database needs to be created dynamically from the various keys.

```
\def_new:Npn \niib_key: {
  \bool_if:NTF \l_niib_before_cite_bool {
    \tlp_use:N \l_niib_key_before_tlp
  }{
    \bool_if:NTF \l_niib_after_cite_bool {
      \tlp_use:N \l_niib_key_after_tlp
    }{
      \tlp_use:N \l_niib_key_mixed_tlp
    }
  }
}
\def_new:Npn \niib_presort: {
  \bool_if:NTF \l_niib_before_cite_bool {
    \tlp_use:N \l_niib_presort_before_tlp
  }{
    \bool_if:NTF \l_niib_after_cite_bool {
      \tlp_use:N \l_niib_presort_after_tlp
    }{
      \tlp_use:N \l_niib_presort_mixed_tlp
    }
  }
}
```

`\niib_convert_endnotes_false:`
`\niib_convert_endnotes_true:`
`\niib_convert_footnotes_false:`
`\niib_convert_footnotes_true:`

Conversion of footnotes and endnotes takes place using four dedicated macros, which can then be called by csname.

```
\def_new:Npn \niib_convert_endnotes_false: {
  \niib_debug:n {Restoring~normal~endnotes}
  \let:NN \endnote \niib_saved_endnote:
  \let:NN \endnotemark \niib_saved_endnotemark:
  \let:NN \endnotetext \niib_saved_endnotetext:
}
\def_new:Npn \niib_convert_endnotes_true: {
  \niib_debug:n {Converting~endnotes~to~bibnotes}
  \let:NN \endnote \bibnote
  \let:NN \endnotemark \bibnotemark
  \let:NN \endnotetext \bibnotetext
}
\def_new:Npn \niib_convert_footnotes_false: {
  \niib_debug:n {Restoring~normal~footnotes}
  \let:NN \footnote \niib_saved_footnote:
  \let:NN \footnotemark \niib_saved_footnotemark:
  \let:NN \footnotetext \niib_saved_footnotetext:
}
\def_new:Npn \niib_convert_footnotes_true: {
  \niib_debug:n {Converting~footnotes~to~bibnotes}
  \let:NN \footnote \bibnote
  \let:NN \footnotemark \bibnotemark
  \let:NN \footnotetext \bibnotetext
}
```

\thanks The \thanks macro has to be replaced by one that uses the original \footnotemark and \footnotetext.

```
\def:Npn \thanks #1 {
  \niib_saved_footnotemark:
  \protected@xdef \@thanks {
    \@thanks
    \protect \niib_saved_footnotetext: [\the\c@footnote] {#1}
  }
}
```

5.6 Finalising

\l_niib_keyname_tlp The name used for the key field is stored a little later. The store is set up here.

```
\tlp_new:N \l_niib_keyname_tlp
```

\niib_biliography:n
\printbibnotes
\niib_after_cite:n Various macro definitions depend on the loading of other packages. When biblatex is in use, life is rather different to when the standard citation method is operational. The \niib_key: macro is used to control sorting: biblatex uses a different macro name for this function than normal LaTeX does. The \printbibnotes user macro always prints the bibliography, using only the notes database if possible. The \niib_after_cite:n macro has to keep a track of the appropriate bibnotes and also fiddle with the \@filesw switch to prevent writing at the wrong time.

```
\AtBeginDocument{
  \err_info:nn {
    (xnotes2bib) \text_put_sp: Beginning~\token_to_str:N
    \AtBeginDocument \text_put_sp: tasks
  } {}
  \@ifpackageloaded {biblatex} {
    \tlp_set:Nn \l_niib_keyname_tlp {keysort}
    \gappto \blx@bibfiles {,\niib_prefix\c_job_name_tlp}
    \let_new:NN \printbibnotes \printbibliography
    \def_new:Npn \niib_after_cite:n #1 {
      \AtNextCite {\@fileswfalse}
      \niib_cite:n {#1}
    }
  }{
    \tlp_gset:Nn \l_niib_keyname_tlp {key}
    \let_new:NN \niib_biliography:n \bibliography
    \def:Npn \bibliography #1 {
      \int_compare:nNnTF {\g_niib_note_int} = {\c_zero} {
        \niib_biliography:n {#1}
      }{
        \niib_biliography:n {
          #1, \l_niib_prefix_tlp \c_job_name_tlp
        }
```

```
    }
  }
  \NewDocumentCommand {\printbibnotes} {} {
    \niib_bibliography:n {\l_niib_prefix_tlp\c_job_name_tlp}
  }
```

`\ifniib@filesw` There is a need to back-up the `\@filesw` switch. As this is a LaTeX 2_ε switch, the name and creation method used here are not LaTeX3-based.

```
  \newif \ifniib@filesw
```

`\@restore@auxhandle` `\niib_auxhandle:` If cite is loaded, there is further trouble.

```
  \@ifpackageloaded {cite}{
    \def_new:Npn \niib_after_cite:n #1 {
       \let:NN \ifniib@filesw \if@filesw
       \@fileswfalse
       \def:Npn \niib_auxhandle: {
         \let:NN \if@filesw \ifniib@filesw
         \let:NN \niib_auxhandle: \scan_stop:
       }
       \niib_cite:n {#1}
    }
    \tlp_gput_right:Nn \g_niib_document_tlp {
      \cs_if_exist:NTF \@restore@auxhandle{
        \tlp_put_right:Nn \@restore@auxhandle {\niib_auxhandle:}
      }{
        \def_new:Npn \@restore@auxhandle {\niib_auxhandle:}
      }
    }
    \let_new:NN \niib_auxhandle: \scan_stop:
```

Life is slightly easier without cite.

```
  }{
    \def_new:Npn \niib_after_cite:n #1 {
       \let:NN \ifniib@filesw \if@filesw
       \@fileswfalse
       \niib_cite:n {#1}
       \let:NN \if@filesw \ifniib@filesw
    }
  }
}
```

`\niib_saved_footnote:` `\niib_saved_footnotemark:` `\niib_saved_footnotetext:` `\niib_saved_endnote:` `\niib_saved_endnotemark:` `\niib_saved_endnotetext:` The second thing to worry about is footnotes and endnotes. The definitions are saved here so that any other packages can have made their changes. No argument specifiers are given as these are saving user space macros.

```
\let_new:NN \niib_saved_footnote: \footnote
```

```
  \let_new:NN \niib_saved_footnotemark: \footnotemark
  \let_new:NN \niib_saved_footnotetext: \footnotetext
  \let_new:NN \niib_saved_endnote: \endnote
  \let_new:NN \niib_saved_endnotemark: \endnotemark
  \let_new:NN \niib_saved_endnotetext: \endnotetext
```

Any conversion can now be applied, before the `convert footnotes` and `convert endnotes` options are redefined to work immediately.

```
  \niib_convert_endnotes:
  \niib_convert_footnotes:
  \keys_manage_quick:n {
    /niib/.cd:,
    convert~endnotes/true/.code:n   = {\niib_convert_endnotes_true:},
    convert~endnotes/false/.code:n  = {\niib_convert_endnotes_false:},
    convert~footnotes/true/.code:n  = {\niib_convert_footnotes_true:},
    convert~footnotes/false/.code:n = {\niib_convert_footnotes_false:},
  }
  \err_info:nn {
    (xnotes2bib) \text_put_sp: Finished~\token_to_str:N
    \AtBeginDocument \text_put_sp: tasks
  } {}
}
```

`\niib_rerun_check:n` If either of the out-of-order options are in use, then a comparison of the citations used with those used in the last run is needed. There are a couple of easy checks to make: if only one list is empty, there must be changes. If both lists are occupied, a check of every item must be made.

```
\def_new:Npn \niib_rerun_check:n #1 {
  \clist_if_empty:cTF {g_niib_old#1cite_clist}{
    \clist_if_empty:cF {g_niib_#1cite_clist}{
      \niib_rerun_log:
    }
  }{
    \clist_if_empty:cTF {g_niib_#1cite_clist}{
      \niib_rerun_log:
```

If both lists are occupied by data, the total contents of each one have to be compared with the other. This needs to be done both ways round.

```
    }{
      \bool_set_false:N \l_tmpa_bool
      \def:Npn \niib_rerun_test:n ##1 {
        \clist_if_in:cnTF {g_niib_old#1cite_clist} {##1} {
          \bool_set_true:N \l_tmpa_bool
          \clist_map_break:w
        }{
        }
      }
```

```
      \clist_map_function:cN {g_niib_#1cite_clist}
        \niib_rerun_test:n
      \def:Npn \niib_rerun_test:n ##1 {
        \clist_if_in:cnTF {g_niib_#1cite_clist} {##1} {
          \bool_set_true:N \l_tmpa_bool
          \clist_map_break:w
        }{
        }
      }
      \clist_map_function:cN {g_niib_old#1cite_clist}
        \niib_rerun_test:n
      \bool_if:NF \l_tmpa_bool {
        \niib_rerun_log:
      }
    }
  }
}
```

`\niib_rerun_test:n` This is redefined in the above, but is made available here.

```
\let_new:NN \niib_rerun_test:n \use_none_i:n
```

`\niib_rerun_log:` The information message needs to be available in a few different places.

```
\def_new:Npn \niib_rerun_log: {
  \err_info:nn {
    (xnotes2bib) \text_put_sp: To~get~notes~in~the~correct~order,~
    please \err_newline: rerun~LaTeX,~(re)run~BibTeX~on~the~file~
    \c_job_name_tlp.aux \err_newline: and~rerun~LaTeX~again~afterwards
  } {(xnotes2bib) \text_put_sp:}
  \io_put_log:x {
    REQ:3:latex:REQ \iow_newline:
    REQ:2:bibtex:REQ \iow_newline:
    REQ:1:latex:REQ \iow_newline:
  }
  \let \niib_rerun_log: \scan_stop:
}
```

At the end of the document, the package should tidy up.

```
\AtEndDocument{
  \niib_rerun_check:n {before}
  \niib_rerun_check:n {after}
  \flushnotestack
  \iow_expanded:Nn \@auxout {
    \token_to_str:N \niibtotalnotes {\int_to_arabic:n \g_niib_note_int}
  }
  \num_compare:nNnF {\g_niib_totalnotes_int} = {\g_niib_note_int} {
    \niib_rerun_log:
```

```
388   }
389   \cs_if_free:NF \g_niib_out_stream {
390     \iow_close:N \g_niib_out_stream
391   }
392 }
393 ⟨/package⟩
```

Change History

v0.0

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.

I

K

L

M

N

O

P

R

S

T

U

W