notes2bib — Integrating notes into the bibliography*

Joseph Wright†

Released 2008/01/08

Abstract

The notes2bib package 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

1 Background

In most subject areas, bibliographic citations and notes are separate entities. However, in some parts of the physical sciences (chemistry and physics) it is usual for references to the literature and notes to be given together in a "References and Notes" section. By default, this requires that BibTeX users create a notes database for each document that they write.

*This file describes version v1.3, last revised 2008/01/08.

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

The endnotes package allows the user to create endnotes rather than footnotes. However, this does not place the notes in the bibliography. The APS have developed the REVTEX document class, which allows footnotes and endnotes to be added to the bibliography. Notes can only be placed at the end of the bibliography using this system. Furthermore, the code to achieve this effect is not available as a package separate from REVTEX.

The aim of the notes2bib package is to make integration of notes into the bibliography easy. Notes can be written as normal in the LATEX source, and are automatically moved to the bibliography. The package is compatible with sorted and unsorted bibliography styles. The package has been designed for use with numerical citations, although it will work with other systems.

2 Basic use

`\bibnote` In the most basic form, the package can be used simply by loading it in the preamble as normal. This adds a new type of note to the existing `\footnote` type: `\bibnote{`⟨*text*⟩`}`. This can be used in exactly the same way as a footnote, taking one mandatory argument ⟨*text*⟩. The ⟨*text*⟩ will be made available as to the bibliography as a note (henceforth referred to as a bibnote).

A very simple example of a bibliography note [1].

```
A very simple example of a bibliography note
\bibnote{Note for the first example}.
```

By default, each bibnote is given an automatically-generated label. However, `\bibnote` accepts an optional argument ⟨*label*⟩, which can be used to over-ride this. This is particularly useful when a note will be referenced several times (The use of the `\citenote` command is covered in Section 2.3).

An example of a named note [2]. The text can then continue and reference the note again later [2].

```
An example of a named note \bibnote[labelled-note]{Note for the
second example}. The text can then continue and reference the note
again later \citenote{labelled-note}.
```

`\bibnotemark` `\bibnotetext` In common with `\footnote`, the basic `\bibnote` macro has companion macros `\bibnotemark` and `\bibnotetext`. The text provided for each not is not "fragile," and so it should not be necessary to use `\bibnotemark` directly. It is needed when replacing footnotes by bibnotes. Notice that there *are* places where bibnotes will be problematic, for example in section headings which also appear in the Table of Contents. In these contexts, use `\citenote` to reference the note, or use an optional argument to the `\section`, *etc.*

It is hard to write a good example for this [3]! The text continues here.

```
It is hard to write a good example for this \bibnotemark! The text
continues here\bibnotetext{Note for the third example}.
```

2.1 Package control

`\niibsetup` The notes2bib package can be controlled using package options, and also dynamically using the `\niibsetup` macro. In both cases the same list of keyval options are recognised, in a similar manner to the graphicx or hyperref packages. Several

of the package options are aimed at controlling the package internally, but by providing a single macro to control this, use is made easier.[1]

Almost all of the package options take literal text; those which do not are true/false switches.

- `cite`: The csname of the macro used to cite bibnotes; by default "cite".
- `endnotes`: Whether to convert endnotes into bibnotes; takes a Boolean value.
- `field`: The BibTeX database field name for notes.
- `footnotes`: Whether to convert footnotes into bibnotes; takes a Boolean value, and does not affect the `\thanks` macro.
- `keyhead`, `keynone`, `keytail`: Sorted BibTeX styles can use the `key` field to sort citations; by setting a prefix to the bibnote name, extra control over sorting can be obtained.
- `log`: The amount of detail to add to the log; expects a value from the list `debug` (very detailed information), `verbose` (the same as `debug`), `normal`, `errors` (errors only), `none`.
- `name`: The name given to bibnote citations; by default, this is followed by an automatically-generated number.
- `prefix`: The file-name prefix used for the BibTeX database holding the notes.
- `presorthead`, `presortnone`, `presorttail`: For biblatex users, the presort field can be used to control sorting of the bibliography; these keys control the value used depending on the setting of the `sort` key.
- `record`: The name of the BibTeX record type used to store notes.
- `sort`: Controls the placement of notes relative to real citations in the bibliography; expected a value `none` (no control of sorting, intended for unsorted bibliographies and interspersed citations and notes), `head` (notes appear before real citations) and `tail` (notes appear after real citations).

The default options are:

[1] Users upgrading from earlier versions of notes2bib will note that the large number of control macros have all been removed from v1.3.

```
\niibsetup{%
  cite=cite,
  endnotes=false,
  field=note,
  footnotes=false,
  keyhead=aaa,
  keynone={},
  keytail=zzz,
  log=normal,
  name=Bibnote,
  prefix=niib-,
  presorthead=ml,
  presortnone=mm,
  presorttail=mn,
  record=Misc,
  sort=none}
```

The `sort` option requires some comment. By default, notes2bib places notes where cited into the `.aux` file, which means that the order of citations and notes depends on the `.bst` file in use. With an unsorted style, citations and notes will be mixed in the order they appear in the LaTeX source. The `sort=head` option will cause notes2bib to place notes before real citations. This should work with sorted and unsorted `.bst` files, but requires two LaTeX runs *before* a BibTeX run in order to work. The package warns if extra runs are needed. The `sort=tail` option places notes after citations; here, only one LaTeX run is needed.

The options `head` and `tail` are provided as shortcuts for `sort=head` and `sort=tail`, respectively. The `debug` option is a shortcut for `log=debug`. As the `endnotes` and `footnotes` options take Boolean values, giving the option name alone is the same as giving ⟨*option*⟩`=true`.

2.2 Output of notes

Bibnotes are only printed when a bibliography is created. This means that at the very least a `\bibliographystyle` command must appear in the source.[2] Under most circumstances, the user will be citing literature, and so will also include a `\bibliography` command in their source. Bibliography notes are automatically added to the citations to be printed.

`\printbibnotes` If bibnotes are being used without any other citations, then the user cannot place `\bibliography` in the source.[3] The package therefore provides the macro `\printbibnotes`, which will output only the notes. If the endnotes package has been loaded, the `\theendnotes` macro is redefined to achieve the same effect.

2.3 Cross-referencing notes

`\citenote` As explained above, each note is automatically assigned a label, or the user can provide one as an optional argument to the note. In either case, notes may then be cross-referenced. Notes are available to be cited directly using the `\cite` command. However, this can cause problems when using the `sort=tail` option.

[2] For biblatex users, the package must be loaded!

[3] LaTeX will complain if the user puts `\bibliography{}`.

The \citenote command is therefore provided. This is aware of the options, and will act correctly in all circumstances.

Cross-references to the note labelled earlier using [2] and using [2].

```
Cross-references to the note labelled earlier
using \cite{labelled-note} and using
\citenote{labelled-note}.
```

2.4 Interaction with other packages

notes2bib is designed to work well with as many other packages as possible. It tries to avoid changing anything which other packages may rely on. That said, it is best to load notes2bib *after* other packages that affect citations, footnotes or endnotes. As usual, the notes2bib package should be loaded before hyperref.[4]

The notes2bib package is compatible with the current release of biblatex (v0.7). As biblatex works very differently from other citation systems, it must be loaded *before* notes2bib. This allows the package to adjust correctly for some important differences in operation of the bibliography environment. Older versions of biblatex *will not work*.

3 Special effects

\flushnotestack

When using the sort=tail option, citations are added to a stack as they are made. This stack is then flushed to the .aux file at the end of the document. If references are given by chapter (or other unit), this may not give the desired effect. The \flushnotestack macro will cause all saved citations to be written at that point, and will reset the stack for continued use. This can therefore be used to control when citation occurs.[5]

\thebibnote

If a sorted bibliography style is in use, and more than nine notes are created, the sort order will be incorrect. This is because by default notes2bib does not pad the automatically-created labels with zeros. To get the correct sort order, \thebibnote should be redefined.

```
\makeatletter
\renewcommand*{\thebibnote}{%
  \niib@name%
  \ifnum\value{bibnote} < 9 0\fi%
  \the\value{bibnote}}
\makeatother
```

4 Known issues

From v1.1, the method for writing notes to the BibTeX database has been modified. This means that bibnotes cannot contain verbatim text.[6] This is the same as for normal footnotes, and so the usual work-arounds are applicable.

[4] It is usually the case that hyperref should be the very last package loaded in the preamble.

[5] This macro was called \flushcitestack prior to v1.3.

[6] Actually, they can, but the spacing will go wrong. LaTeX will only complain if a note ends with verbatim text. However, verbatim text is not supported in bibnotes: don't do it!

The next note contains some awkward text [4].

```
The next note contains some awkward text
\bibnote{Some \texttt{\textbackslash verb}-like output}.
```

5 The mechanism

The mechanism for positioning notes in the bibliography is somewhat involved. Rather than expect interested users to read all of the code that follows, a condensed overview is given here. The thinking behind the system used is explained first, by considering the endnotes package and REVTEX class. Both of these provided inspiration for this package.

5.1 The `endnotes` approach

The endnotes package[7] allows the user to generate endnotes in the same way as footnotes. In endnotes, the text of the note is written to a `.ent` file. This is achieved in an unexpanded form using the `\meaning` TEX primitive. To produce the list of endnotes, this file is read back into LATEX, with the extra information `\meaning` also writes being stripped off in the process.

This method is relativity simple in concept, but obviously does not integrate with BIBTEX. The use of `\meaning` for unexpanded output also means that information requires further processing before it can be included in the bibliography.

5.2 The REVTEX approach

REVTEX takes a similar approach to creating endnotes, but also allows footnotes to be converted into endnotes. This results in a file containing all of the non-literature citations in a document in a single external file (in this case a `.end` file). REVTEX also uses a different method to achieve unexpanded output, meaning that several macros are not "active" in notes.

The second part of the REVTEX approach is to (optionally) read the notes back into the document. This is achieved by modifying the `\bibliography` environment to output each note in the bibliography. This takes place *en masse*, after the normal citations.

REVTEX makes a number of modifications to LATEX, and is dependent on using natbib. The method used is also not compatible with interspersing normal citations and notes.

5.3 The `notes2bib` approach

In notes2bib, notes are again written to an external file. However, in contrast to the methods those outlined above, notes2bib writes its output in the well-known BIBTEX database format. All of the note text is written almost completely unexpanded to the file, the only requirement being that the braces match within the argument.[8]

Each note results in a citation being placed by notes2bib in the `.aux` file. The `\bibliography` command is also modified so that the new database will

[7] http://tug.ctan.org/macros/latex/contrib/misc/endnotes.sty

[8] Writing to the file uses the ε-TEX `\unexpanded` primitive.

be used by BibTeX. After the BibTeX run, the note text will appear in the `.bbl` file, in the same way as any other citation. Using an unsorted BibTeX style, this results in notes interspaced with the normal citations. For sorted styles, notes2bib allows various methods for controlling the placement of notes, based on writing appropriate fields in the BibTeX database.

`niibheadcite` The `sort=head` option works by adding an additional macro to the `.aux` file: `\niibheadcite`. On the next LaTeX run, this causes the relevant notes to be cited right at the beginning of the document, before any real citations.[9] For `sort=tail`, the value of `\if@filesw` is temporarily altered to prevent writing of a citation to the `.aux` file. The citation is added to a stack, and is written at the end of the document.

6 The package code

6.1 Package setup code

The package starts with the usual identification code.

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{notes2bib}
  [2008/01/08 v1.3 Integrating notes into the bibliography]
```

The package requires ε-TeX, so before going any further, this is tested. This code is taken more-or-less verbatim from biblatex.

```
\begingroup
  \@ifundefined{eTeXversion}
    {\PackageError{notes2bib}
       {Not running under e-TeX}
       {This package requires e-TeX. Try compiling the document
        with\MessageBreak 'elatex' instead of 'latex'. When using
        pdfTeX, try 'pdfelatex'\MessageBreak instead of 'pdflatex'}
     \endgroup\endinput}
    {\endgroup}
```

The necessary support packages are loaded.

```
\RequirePackage{xkeyval,etoolbox}
```

`\niib@tempa` `\niib@tempb` Some private temporary macros are declared.

```
\newcommand*{\niib@tempa}{}
\newcommand*{\niib@tempb}{}
```

6.2 Logging

`\ifniib@debug` `\ifniib@logmin` `\ifniib@lognone` To control logging, some new switches are declared.

```
\newif\ifniib@debug
\newif\ifniib@logmin
\newif\ifniib@lognone
```

`\niib@log@err` `\niib@log@warn` `\niib@log@inf` Some handy re-usable macros are defined here. These all take names beginning These pop up in various places. First errors, warnings and information are handled. Package options are used to control how much output is given.

[9] Thanks to Michael Shell for the idea for this method.

```
\newcommand*{\niib@log@err}[2]{%
  \ifniib@lognone\else
    \ifniib@logmin
      \PackageWarning{notes2bib}{#1}%
    \else
      \PackageError{notes2bib}{#1}{#2}%
    \fi
  \fi}
\newcommand*{\niib@log@warn}[1]{%
  \ifniib@lognone\else
    \ifniib@logmin\else
      \PackageWarning{notes2bib}{#1}%
    \fi
  \fi}
\newcommand*{\niib@log@inf}[1]{%
  \ifniib@lognone\else
    \ifniib@logmin\else
      \PackageInfo{notes2bib}{#1}%
    \fi
  \fi}
```

`\niib@log@debug` The debug macro only gives output if the appropriate package option is set.

```
\newcommand*{\niib@log@debug}[1]{%
  \ifniib@lognone\else
    \ifniib@debug
      \PackageInfo{notes2bib}{#1}%
    \fi
  \fi}
```

6.3 Package options

`\niib@opt@boolkey` To aid maintenance, some shortcuts are defined for generating keys. These also allow the debugging messages to be added automatically to every key.

```
\newcommand*{\niib@opt@boolkey}[2][]{%
  \define@boolkey[niib]{opt}[niib@]{#2}[true]
  {#1\niib@log@debug{Option #2 set to ##1}}}
```

`\niib@opt@choicekey` A "fill in the blanks" choice key. In all cases, `\niib@tempa` is used to hold the value given to the key, so that `\ifx` testing can occur.

```
\newcommand*{\niib@opt@choicekey}[5][]{%
  \define@choicekey*+[niib]{opt}{#2}[\niib@tempa]{#3}[#1]
    {#4\niib@log@debug{Option #2 set to ##1}}
    {#5\niib@log@debug{Option #2 set to ##1}}}
```

`\niib@opt@cmdkeys` A shortcut for xkeyval command keys.

```
\newcommand*{\niib@opt@cmdkeys}[1]{%
  \define@cmdkeys[niib]{opt}[niib@]{#1}}
```

`\niibsetup` To allow modification of options at run time, a setup macro is provided. The run of strange tests are to prevent problems in arrays and the like.

```
\newcommand*{\niibsetup}[1]{%
  \iffalse{\fi\ifnum0=`}\fi
  \setkeys[niib]{opt}{#1}%
```

```
\ifnum0='{\fi\iffalse}\fi}
```

`\niib@opt@log` `\niib@tempa` `\niib@tempb`

The xkeyval package option for logging is declared. This is then processed to set the switches correctly.

```
\niib@opt@choicekey[normal]{log}{debug,verbose,normal,errors,none}
```

A series of comparisons are made to assign the logging mode. The `normal` option is not tested, as executing the option sets the switches appropriately.

```
 {\niib@debugfalse
  \niib@logminfalse
  \niib@lognonefalse
  \renewcommand*{\niib@tempb}{none}%
  \ifx\niib@tempa\niib@tempb
    \niib@lognonetrue
  \fi
  \renewcommand*{\niib@tempb}{minimal}%
  \ifx\niib@tempa\niib@tempb
    \niib@logmintrue
  \fi
  \renewcommand*{\niib@tempb}{debug}%
  \ifx\niib@tempa\niib@tempb
    \niib@debugtrue
  \fi
  \renewcommand*{\niib@tempb}{verbose}%
  \ifx\niib@tempa\niib@tempb
    \niib@debugtrue
  \fi}
```

The option has not been recognised: give a warning (if appropriate).

```
  {\niib@log@warn{Unrecognised value '#1' for option log}}
```

`\niib@opt@debug`

A quick method to set `log=debug`.

```
\niib@opt@boolkey{debug}
```

`\niib@opt@footnotes` `\niib@opt@ndnotes` `\ifniib@footnotes` `\ifniib@endnotes`

The `footnote` and `endnote` options are declared here.

```
\niib@opt@boolkey[\niib@swapfoot]{footnotes}
\niib@opt@boolkey[\niib@swapend]{endnotes}
```

`\ifniib@head` `\ifniib@tail`

Switches are needed for placing notes before and after normal citations.

```
\newif\ifniib@tail
\newif\ifniib@head
```

`\niib@opt@sort` `\niib@tempa` `\niib@tempb`

This option controls the position of notes *versus* normal citations. The xkeyval option replaces the earlier `head` and `tail` options, which are retained for backward compatibility.[10]

```
\niib@opt@choicekey[none]{sort}{none,head,tail}
 {\niib@headfalse
  \niib@tailfalse
  \renewcommand*{\niib@tempb}{head}%
  \ifx\niib@tempa\niib@tempb
    \niib@headtrue
  \fi
```

[10] This may change in a future release; the options `head` and `tail` are "depreciated."

```
    \renewcommand*{\niib@tempb}{tail}%
    \ifx\niib@tempa\niib@tempb
      \niib@tailtrue
    \fi}
  {\niib@log@warn{Unrecognised value `#1' for option sort}}
```

\niib@opt@head
\niib@opt@tail

The back-compatibility code; unlike earlier versions, this will take whatever the `sort`-type key is given.

```
\niib@opt@boolkey[%
  \ifniib@head
    \ifniib@tail
      \niib@tailfalse
      \niib@log@inf{Option head cancels existing\MessageBreak
        tail or sort=tail option}
    \fi
  \fi]{head}
\niib@opt@boolkey[%
  \ifniib@tail
    \ifniib@head
      \niib@headfalse
      \niib@log@inf{Option tail cancels existing\MessageBreak
        head or sort=head option}
    \fi
  \fi]{tail}
```

\niib@opt@cite
\niib@cite
\niib@opt@name
\niib@name
\niib@opt@prefix
\niib@prefix
\niib@opt@record
\niib@record
\niib@opt@field
\niib@field
\niib@opt@presorthead
\niib@presorthead
\niib@opt@presortnone
\niib@presortnone
\niib@opt@presorttail
\niib@presorttail
\niib@opt@keyhead
\niib@keyhead
\niib@opt@keynone
\niib@keynone
\niib@opt@keytail
\niib@keytail

The various internal control values are set up as command keys.

```
\niib@opt@cmdkeys{%
  cite,
  name,
  prefix,
  record,
  field,
  presorthead,
  presortnone,
  presorttail,
  keyhead,
  keynone,
  keytail}
\niibsetup{%
  cite=cite,
  name=Bibnote,
  prefix=niib-,
  record=Misc,
  field=note,
  presorthead=ml,
  presortnone=mm,
  presorttail=mn,
  keyhead=aaa,
  keynone={},
  keytail=zzz}
```

6.4 Footnote and endnote handling

\niib@org@footnote
\niib@org@footnotemark
\niib@org@footnotetext
\niib@org@thanks

To allow dynamic handling of footnotes and endnotes, the original definitions are backed up. This is done at the start of the document, so that changes from any other packages are picked up.

```
\AtBeginDocument{%
  \let\niib@org@footnote\footnote
  \let\niib@org@footnotemark\footnotemark
  \let\niib@org@footnotetext\footnotetext
  \let\niib@org@thanks\thanks
```

\niib@org@endnote
\niib@org@endnotemark
\niib@org@endnotetext
\niib@org@theendnotes

If endnotes is loaded, then \endnote and friends have to be saved.

```
  \@ifpackageloaded{endnotes}
    {\let\niib@org@endnote\endnote
     \let\niib@org@endnotemark\endnotemark
     \let\niib@org@endnotetext\endnotetext
     \let\niib@org@theendnotes\theendnotes}}
```

\niib@thanks

The first step of converting footnotes into bibnotes is to patch the \thanks macro so that it uses the original definitions of \footnotemark and \footnotetext. The necessary setup is also put in place for on-the-fly swapping of footnotes and bibnotes.

```
\AtBeginDocument{%
  \let\niib@thanks\thanks
  \let\niib@footnotetext\footnotetext
  \let\niib@footnotemark\footnotemark
  \patchcmd{\niib@thanks}{\footnotetext}{\niib@org@footnotetext}
    {\patchcmd{\niib@thanks}{\footnotemark}
      {\niib@org@footnotemark}
```

\thanks
\niib@footnotetext
\niib@footnotemark

If the package gets here, then both replacements have worked. The updated definition of \thanks is applied (it does not depend on package options), and the definitions of \footnotemark and \footnotetext that are needed are set up.

```
      {\let\thanks\niib@thanks
       \let\niib@footnotetext\bibnotetext
       \let\niib@footnotemark\bibnotemark
       \niib@log@debug{Successfully patched \noexpand\thanks}}
```

This group deals with a failure of the second patch.

```
      {\niib@log@warn{Failed to patch \noexpand\thanks\MessageBreak
         \noexpand\footnotemark\space and
         \noexpand\footnotetext\MessageBreak are unmodified}%
       \niib@log@debug{Could not substitute
         \noexpand\footnotemark\MessageBreak in \noexpand\thanks}}}
```

If the first patch fails, then the error shows up here.

```
    {\niib@log@warn{Failed to patch \noexpand\thanks\MessageBreak
       \noexpand\footnotemark\space and
       \noexpand\footnotetext\MessageBreak are unmodified}%
     \niib@log@debug{Could not substitute
       \noexpand\footnotetext\MessageBreak in \noexpand\thanks}}}
```

\niib@swapfoot
\footnote
\footnotemark
\footnotetext

The swapping code can now be implemented.

```
\newcommand*{\niib@swapfoot}{}
\AtBeginDocument{%
  \renewcommand*{\niib@swapfoot}{%
    \ifniib@footnotes
      \let\footnote\bibnote
      \let\footnotemark\niib@footnotemark
      \let\footnotetext\niib@footnotetext
      \niib@log@debug{Converting footnotes to bibnotes}%
    \else
      \let\footnote\niib@org@footnote
      \let\footnotemark\niib@org@footnotemark
      \let\footnotetext\niib@org@footnotetext
      \niib@log@debug{Using kernel definition of footnotes}%
    \fi}
  \niib@swapfoot}
```

\niib@swapend
\endnote
\endnotemark
\endnotetext
\theendnotes

For endnotes, the code needed depends on whether the endnotes package is available or not. If it is, then swapping the two definitions is set up.

```
\newcommand*{\niib@swapend}{}
\AtBeginDocument{%
  \@ifpackageloaded{endnotes}
    {\renewcommand*{\niib@swapend}{%
       \ifniib@endnotes
         \let\endnote\bibnote
         \let\endnotemark\bibnotemark
         \let\endnotetext\bibnotetext
         \let\theendnotes\printbibnotes
         \niib@log@debug{Converting endnotes to bibnotes}%
       \else
         \let\endnote\niib@org@endnote
         \let\endnotemark\niib@org@endnotemark
         \let\endnotetext\niib@org@endnotetext
         \let\theendnotes\niib@org@theendnotes
         \niib@log@debug{Using endnotes package to handle endnotes}%
       \fi}
     \niib@swapend}
```

endnotes is not loaded; once the `endnotes` option has been given, there is nothing to go back. Of course, if the user does not give the `endnotes` option, `\endnote` is not defined at all.

```
    {\ifniib@endnotes
       \let\endnote\bibnote
       \let\endnotemark\bibnotemark
       \let\endnotetext\bibnotetext
       \let\theendnotes\printbibnotes
       \niib@log@debug{Converting endnotes to bibnotes}%
     \fi
     \renewcommand*{\niib@swapend}{%
       \ifniib@endnotes
         \let\endnote\bibnote
         \let\endnotemark\bibnotemark
         \let\endnotetext\bibnotetext
         \let\theendnotes\printbibnotes
```

```
213         \niib@log@debug{Converting endnotes to bibnotes}%
214       \else
215         \niib@log@inf{endnotes package not loaded\MessageBreak
216           endnotes=false ignored}%
217       \fi}}}
```

6.5 User macros

\thebibnote A counter is needed for the notes created. In analogy to other counters in LaTeX, this is given a `\the...` name. The user should not really need to use this macro, but convention dictate that it has a user-space name. The LaTeX `\newcounter` macro is used (rather than the TeX `\newcount`) as the automatic system expects the numbers to be globally unique.

```
218 \newcounter{bibnote}
219 \renewcommand*{\thebibnote}{\niib@name\the\value{bibnote}}
```

\bibnote Each new bibnote increments the note counter, then checks for an optional label, before handing off to the internal macro `\niib@bibnote`.

```
220 \DeclareRobustCommand*{\bibnote}{%
221   \stepcounter{bibnote}%
222   \@ifnextchar[%]
223     {\niib@bibnote}
224     {\niib@bibnote[\thebibnote]}}
```

\bibnotemark The `\bibnotemark` macro works in the same way as `\bibnote`, but calls `\niib@mark` rather than `\niib@bibnote`.

```
225 \DeclareRobustCommand*{\bibnotemark}{%
226   \stepcounter{bibnote}%
227   \@ifnextchar[%]
228     {\niib@mark}
229     {\niib@mark[\thebibnote]}}
```

\bibnotetext The text companion to the mark macro above, with no increment of the counter. There is nothing special to do, so the LaTeX kernel handling of optional arguments can be used.

```
230 \DeclareRobustCommand*{\bibnotetext}[1][\thebibnote]{%
231   \niib@text{#1}}
```

\printbibnotes To allow for the possibility of there being no other notes, a command to print only notes is given. In the biblatex case, the best that can be done is to issue `\printbibliography`.

```
232 \@ifpackageloaded{biblatex}
233   {\let\printbibnotes\printbibliography}
234   {\DeclareRobustCommand*{\printbibnotes}
235      {\niib@org@bib{\niib@prefix\jobname}}}
```

\flushnotestack In order to delay citations to the end of the bibliography (and thus force others to the start), a "stack" is created of citations which need to be written to the `.aux` file. This is done here, and the stack is cleared so collection can begin again.

```
236 \DeclareRobustCommand*{\flushnotestack}{%
237   \ifx\@empty\niib@stack\@empty
238     \niib@log@debug{Citation stack empty: nothing for\MessageBreak
```

```
      \noexpand\flushnotestack to do}%
  \else%
    \niib@log@debug{Flushing note citations to aux file}%
    \expandafter\nocite\expandafter{\niib@stack}%
    \gdef\niib@stack{}%
  \fi}
```

`\citenote` Problems arise with `\cite` and the `sort=tail` option. Rather than overload `\cite` with all of the problems that can bring, a new command is provided that can be guaranteed to work.

```
\DeclareRobustCommand*{\citenote}[1]{\niib@mark[#1]}
```

6.6 Internal macros

`\niib@keyname` If biblatex is in use, the key field in the BibTeX database should be called "keysort," whereas otherwise it should be "key."

```
\AtBeginDocument{%
  \@ifpackageloaded{biblatex}
    {\niib@log@debug{Using field 'keysort' for sorting key}%
     \newcommand*\niib@keyname{keysort}}
    {\niib@log@debug{Using field 'key' for sorting key}%
     \newcommand*\niib@keyname{key}}}
```

`\niib@presort` `\niib@key` The values taken by `\niib@presort` and `\niib@key` depend on the desired positioning of notes in the bibliography.

```
\newcommand*{\niib@presort}{%
  \ifniib@head
    \niib@presorthead%
  \else
    \ifniib@tail
      \niib@presorttail%
    \else
      \niib@presortnone%
    \fi
  \fi}
\newcommand*{\niib@key}{%
  \ifniib@head
    \niib@keyhead%
  \else
    \ifniib@tail
      \niib@keytail%
    \else
      \niib@keynone%
    \fi
  \fi}
```

`\niib@msg` To inform the user, the automatically-created BibTeX database needs to carry suitable information on its source.

```
\edef\niib@msg{%
  This is an auxiliary file used by the 'notes2bib' package.^^J%
  This file may safely be deleted. It will be recreated as
  required.^^J}
```

\niib@stack This macro is needed to store any citations at the end of the bibliography. Initially, this is empty.

```
\newcommand*{\niib@stack}{}
```

\niib@addtostack \niib@tempa The various optional argument tricks above all use the same core code, which adds the mandatory argument of the citation to the stack. The stack is global (see also \flushnotestack).

```
\newcommand*{\niib@addtostack}[1]{%
  \niib@log@debug{Adding citation #1 to stack}%
  \edef\niib@tempa{#1}%
  \ifx\@empty\niib@stack\@empty
    \xdef\niib@stack{\niib@tempa}%
  \else
    \xdef\niib@stack{\niib@stack,\niib@tempa}%
  \fi}
```

\niib@bibnote Two steps are needed here, writing the text of the note to file (handled by \niib@text, and marking the citation (using \niib@cite).

```
\long\def\niib@bibnote[#1]#2{%
  \niib@text{#1}{#2}%
  \niib@mark[#1]}
```

\niib@headlist To inform the user that a re-run of LaTeX is needed, tracking is needed of any "head" citations.

```
\newcommand*{\niib@headlist}{}
```

\niib@mark \niib@tempa Adding a citation to the LaTeX file is handled here. When using the sort=head option, the citation is written to the .aux file for sorting control. The normal citation command is then called.

```
\def\niib@mark[#1]{%
  \ifniib@head
    \edef\niib@tempa{#1}%
    \ifx\@empty\niib@headlist\@empty
      \xdef\niib@headlist{\niib@tempa}%
    \else
      \xdef\niib@headlist{\niib@headlist,\niib@tempa}%
    \fi
    \if@filesw
      \niib@log@debug{Adding citation #1 to list for next run}%
      \immediate\write\@auxout{\string\niibheadcite{#1}}%
    \fi
  \fi
```

When the sort=tail option is active, citation is handled by another macro, so a switch is needed.

```
  \ifniib@tail
    \expandafter\niib@tailcite%
  \else
    \expandafter\niib@normcite%
  \fi
  {#1}}
```

\ifniib@filesw A switch is used to back up \if@filesw.

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

\niib@tailcite When using the sort=tail option, bibnote citation need to be stored for later. With biblatex, the \AtEndCite macro is available to provide a hook for the necessary switch. In other cases, the current value of \if@filesw is then saved, before turning it off and setting up the restore system.

```
\AtBeginDocument{%
  \@ifpackageloaded{biblatex}
    {\newcommand{\niib@tailcite}[1]{%
       \niib@addtostack{#1}%
       \AtNextCite{\@fileswfalse}%
       \niib@normcite{#1}}}
    {\newcommand{\niib@tailcite}[1]{%
       \niib@addtostack{#1}%
       \let\ifniib@filesw\if@filesw
       \@fileswfalse
       \let\niib@auxhook\niib@restorefilesw
       \niib@tcite{#1}}}}
```

\niib@restorefilesw Restoring the switch is set up here. The reference to \niib@auxhook ensures that the mechanism is turned off for the next real citation.

```
\newcommand*{\niib@restorefilesw}{%
  \let\if@filesw\ifniib@filesw
  \let\niib@auxhook\relax}
```

\niib@tcite Actually carrying out the citation, and restoring the value of \if@filesw depends on whether cite is loaded.

```
\AtBeginDocument{%
  \@ifpackageloaded{cite}
    {\newcommand*{\niib@tcite}[1]{\niib@normcite{#1}}}
    {\newcommand*{\niib@tcite}[1]{%
       \niib@normcite{#1}%
       \niib@restorefilesw}}
```

\niib@normcite The normal citation command.

```
\newcommand*{\niib@normcite}{\@nameuse{\niib@cite}}
```

\niib@text The "business end" of writing the notes to file. This is a \long macro, so no star is used for \newcommand.

```
\newcommand{\niib@text}[2]{%
```

\niib@out \niib@stream If this is the first note, then a new output stream is needed, otherwise it will already be open.

```
  \@ifundefined{niib@out}{%
    \if@filesw
      \newwrite\niib@out%
      \gdef\niib@stream{\niib@prefix\jobname.bib}%
      \niib@log@debug{Creating BibTeX database file \MessageBreak
        \niib@stream\space to contain bibnotes}%
      \immediate\openout\niib@out\niib@stream\relax
```

The new file starts with the message that it has been automatically generated by notes2bib.

```
      \immediate\write\niib@out{\niib@msg}%
    \fi}{}%
```

The new record is now written to file. The `\unexpanded` ε-TEX primitive is used to avoid expansion of macros in the note text. The only issue with this is the addition of spaces after command names; this is the reason verbatim text cannot be used in bibnotes.

```
\if@filesw
  \niib@log@debug{Writing bibnote #1 contents
    \MessageBreak---\MessageBreak#2\MessageBreak---\MessageBreak
    to BibTeX database}%
  \immediate\write\niib@out{%
    @\niib@record\string{#1,^^J%
    presort = \string{\niib@presort\string},^^J%
    \niib@keyname\space= \string{\niib@key#1\string},^^J%
    \niib@field\space= \string{\unexpanded{#2}\string}^^J%
    \string}^^J}%
\fi}
```

`\niib@headcitelist` To inform the user that a re-run of LATEX is needed, tracking is needed of any citations that have been moved to the start of the `.aux` file. This needs an initially-empty macro.

```
\newcommand*{\niib@headcitelist}{}
```

`\document` `\niib@dochook` `\niibheadcite` `\niib@tempa` When using the `sort=head` option, bibnotes need to appear in the `.aux` file before other citations. Other approaches cause all sorts of problems, so the suggestion of Michael Shell is implemented here. When `head` is active, `\niibheadcite` is added to the `.aux` file. At the next LATEX run, this will add a citation to the beginning of the `.aux` file. To get the `\nocite` to work, a hook has to be added to `\document`. The reason is that `\AtBeginDocument` cannot be used here: it is not available once the old `.aux` file has been read.

```
\g@addto@macro{\document}{\niib@dochook}
\newcommand*{\niibheadcite}[1]{%
  \edef\niib@tempa{#1}%
  \ifx\@empty\niib@headcitelist\@empty
    \xdef\niib@headcitelist{\niib@tempa}%
  \else
    \xdef\niib@headcitelist{\niib@headcitelist,\niib@tempa}%
  \fi
  \if@filesw
    \niib@log@debug{Adding citation #1 to start of .aux file}%
  \fi
  \g@addto@macro{\niib@dochook}{\nocite{#1}}}
```

`\niib@auxhook` `\niib@dochook` `\@restore@auxhandle` To allow the automatic punctuation-searching of cite to work, some code has to be added to the hook available there. However, as that is intended for multibib, care is needed to get the desired result. `\niib@dochook` is defined here, even though it is needed above, as the code here will always be executed.

```
\newcommand*{\niib@dochook}{%
  \@ifundefined{@restore@auxhandle}
    {\newcommand*{\@restore@auxhandle}{\niib@auxhook}}
    {\ifx\relax\@restore@auxhandle\relax
       \newcommand*{\@restore@auxhandle}{\niib@auxhook}%
     \else
       \g@addto@macro{\@restore@auxhandle}{\niib@auxhook}%
```

```
    \fi}}
\newcommand*{\niib@auxhook}{}
\let\niib@auxhook\relax
```

\blx@bibfiles The \bibliography macro is patched to ensure that when it is executed the note file is also processed. biblatex does things very differently, but this actually makes it much easier to patch for.

```
\AtEndPreamble{%
  \@ifpackageloaded{biblatex}%
    {\expandafter\gappto\expandafter\blx@bibfiles\expandafter%
       {,\niib@prefix\jobname}
     \niib@log@debug{Added bibnotes database to biblatex file list}}%
```

\niib@org@bib \bibliography Without biblatex, the bibliography command is patched so that it will run on the automatically-generated BibTeX database. If no notes have been added, then the macro doesn't actually do anything.

```
    {\let\niib@org@bib\bibliography
     \renewcommand{\bibliography}[1]{%
       \ifnum\the\value{bibnote} > \z@
         \niib@org@bib{\niib@prefix\jobname,#1}%
       \else
         \niib@org@bib{#1}%
       \fi}
     \niib@log@debug{Added bibnote database to
       \noexpand\bibliography}}}
```

6.7 Finalisation

\ifniib@rerun A switch is needed for the re-run test.

```
\newif\ifniib@rerun
```

\niib@checkrerun Any "head" notes may mean a second LaTeX run is needed.

```
\newcommand*{\niib@checkrerun}{%
  \niib@rerunfalse
  \ifx\@empty\niib@headlist\@empty
    \ifx\@empty\niib@headcitelist\@empty
      \niib@log@debug{No 'head' notes detected}%
    \else
      \niib@reruntrue
      \niib@log@debug{No 'head' notes found this run\MessageBreak
        but .aux files contained the 'head' requests:\MessageBreak
        \niib@headcitelist}%
    \fi
  \else
    \ifx\@empty\niib@headcitelist\@empty
      \niib@reruntrue
      \niib@log@debug{No 'head' requests in .aux file\MessageBreak
        but 'head' notes in this run:\MessageBreak
        \niib@headlist}%
    \else
```

If the package gets here, then there are some head notes and some head requests in the aux file. The two lists are now compared.

```
      \niib@checklists%
    \fi
  \fi
  \ifniib@rerun
```

Both human-readable requests for new runs, and biblatex-style automated requests are made. The package does not know if BibTeX8 is in use, so just asks for BibTeX.

```
    \niib@log@warn{Rerun LaTeX to get correct `head' notes}%
    \niib@log@warn{Please (re)run BibTeX on the file(s):
      \MessageBreak\jobname.aux
      \MessageBreak and rerun LaTeX afterwards.}%
    \ifniib@lognone\else
      \typeout{%
        REQ:3:latex:REQ^^J%
        REQ:2:bibtex:REQ^^J%
        REQ:1:latex:REQ}%
    \fi
  \fi}
```

\niib@checklists
\niib@tempa
\niib@tempb

The business end of comparing the two lists. Two sweeps are made, to check that the lists match entirely.

```
\newcommand*{\niib@checklists}{%
  \@for\niib@tempa:=\niib@headlist\do{%
    \niib@reruntrue
    \@for\niib@tempb:=\niib@headcitelist\do{%
      \ifx\niib@tempa\niib@tempb
        \niib@rerunfalse
      \fi}
    \ifniib@rerun
      \niib@log@debug{Note \niib@tempa is a `head' note
        \MessageBreak but request not in .aux file}%
    \fi}
  \ifniib@rerun\else
    \@for\niib@tempa:=\niib@headcitelist\do{%
      \niib@reruntrue
      \@for\niib@tempb:=\niib@headlist\do{%
        \ifx\niib@tempa\niib@tempb
          \niib@rerunfalse
        \fi}
      \ifniib@rerun
        \niib@log@debug{Note \niib@tempa is set to `head' in
          .aux\MessageBreak file but is not a `head' note}%
      \fi}
  \fi}
```

At the end of the document, any delayed citations are written to the `.aux` file, and the database file is closed cleanly. A check is also made for the need for an additional LaTeX run for "head" notes.

```
\AtEndDocument{%
  \flushnotestack%
  \niib@checkrerun%
  \@ifundefined{niib@out}{}
    {\immediate\closeout\niib@out%
```

```
451      \niib@log@debug{Closed BibTeX database file\MessageBreak
452        \niib@stream}}}
```

Options are processed at the end of the package, to avoid any odd issues arising with definition of macros.

```
453 \ProcessOptionsX[niib]<opt>
```

7 Change History

8 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

9 Notes

[1] Note for the first example.

[2] Note for the second example.

[3] Note for the third example.

[4] Some \verb-like output.