notes2bib — Integrating notes into the bibliography*

Joseph Wright†

Released 2008/07/22

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 v1.4a, last revised 2008/07/22.

†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]

footnote
endnotes

Some options control the general behaviour of notes2bib in the body of the LaTeX source. The `footnote` and `endnotes` options control whether `\footnote` and `\endnote` macros are converted into bibnotes. Both are Boo-

cite

lean options, and are **`false`** by default. The citation command used by notes2bib to insert bibnotes. It should be the name of a LaTeX command (a "csname"), and is set to **`cite`** on loading notes2bib; this means that `\cite` will be used as the citation command.

field
name
record

A number of options control the data added to the BibTeX database. The `field` and `record` options control the type of BibTeX entry created by notes2bib. On loading, `record` is set to **`Misc`** an `field` is set as **`note`**. Depending on the BibTeX style in use, better choices may exist for these settings. The `name` option is used to automatically generate citation names. The option starts with the value

prefix

`Bibnote`, which may need to change for author–year styles in particular. The name of the database itself is controlled by the `prefix` option. This contains the "marker" used by notes2bib to attach to the job name when creating the storage database. The default is **`niib-`**.

sort
head
tail
keyhead
keynone
keytail
presorthead
presortnone
presorttail

Bibnotes can be created so that thy will be sorted before or after normal citations. A list of values are recognised: **`none`** (no control of sorting), `head` (notes appear before real citations) and `tail` (notes appear after real citations). The shortcut options `head` and `tail` are also available. A number of mechanisms are used to ensure correct sorting of bibnotes. For normal BibTeX users, the options `keyhead`, `keynone` and `keytail` are used to control sorting. These values are added to the start of the citation name in the `key` field, which controls sorting. The default values are **`aaa`**, nothing and **`zzz`**, respectively. For biblatex users, control is made using the `presort` system made available there. The notes2bib options `presorthead`, `presortnone` and `presorttail` set up the appropriate values; default values are **`ml`**, **`mm`** and **`ml`**, respectively.

log
debug
etex

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` (what it says). As a shortcut, the `debug` option is provided as an alias to `log=debug`. The package has a single load-time only option, `etex`. This is a Boolean switch, and determines whether ε-TeX extensions are used if available. This is **`true`** by default; the intension here is for testing when sending to publishers, *etc.*, where ε-TeX may be an issue.

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

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

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

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

\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. 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 has been tested with cite, natbib, hyperref and mciteplus with no problems. The notes2bib package is compatible with the current release of biblatex (v0.7); older versions of biblatex may or may not work.[4]

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] As biblatex is experimental and is not currently added to TEX distributions, users have little excuse for not using the latest release.

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

4 Package requirements

The package has only one requirement, xkeyval version 2.5 or later. ε-TEX is used if available, but is not a requirement.

5 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.

The next note contains some awkward text [4].

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

The package relies on BIBTEX being able to open and process the temporary database containing the note text. The name of this file contains `\jobname`, the name of the main LATEX file being processed. This must consist only of characters that BIBTEX can handle. In particular, spaces in the file name will lead to problems.

6 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.

6.1 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.

6.2 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.

[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!

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

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.

6.3 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 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.

7 The package code

7.1 Package setup code

`\niib@id` The package starts with the usual identification code and support loading.

```
\NeedsTeXFormat{LaTeX2e}
\def\niib@id$#1: #2.#3 #4 #5-#6-#7 #8 #9${%
    #5/#6/#7\space v1.4a\space}
\ProvidesPackage{notes2bib}
  [\niib@id$Id: notes2bib.dtx 13 2008-07-22 20:06:39Z joseph $
   Integrating notes into the bibliography]
\RequirePackage{xkeyval}[2005/05/07]
```

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

`\niib@tempb`

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

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

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

7.2 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.

```
\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}
```

7.3 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}{\setkeys[niib]{opt}}
```

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}
```

\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
```

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.

```
\niib@opt@choicekey[none]{sort}{none,head,tail}
```

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

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@cite
\niib@name
\niib@prefix
\niib@record
\niib@field
\niib@presorthead
\niib@presortnone
\niib@presorttail
\niib@keyhead
\niib@keynone
\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}
```

\ifniib@etex

The option to turn off ε-TEX needs to ensure that the user does not ask for ε-TEX if it is not there. On the other hand, the default is to use ε-TEX if it is available.

```
\niib@opt@boolkey{etex}
\begingroup
  \@ifundefined{eTeXversion}
    {}
    {\global\niib@etextrue}
\endgroup
```

7.4 Footnote and endnote handling

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

To allow dynamic handling of footnotes and endnotes, the original definitions are backed up.

```
\let\niib@org@footnote\footnote
\let\niib@org@footnotemark\footnotemark
\let\niib@org@footnotetext\footnotetext
```

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

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

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

\thanks

The \thanks macro is redefined.

```
\@ifundefined{thanks}{}
  {\renewcommand*{\thanks}[1]{%
     \niib@org@footnotemark
     \protected@xdef\@thanks{\@thanks
     \protect\niib@org@footnotetext[\the\c@footnote]{#1}}}}
```

\niib@swapfoot
\footnote
\footnotemark
\footnotetext

The swapping code can now be implemented.

```
\newcommand*{\niib@swapfoot}{%
  \ifniib@footnotes
    \let\footnote\bibnote
    \let\footnotemark\bibnotetext
    \let\footnotetext\bibnotemark
    \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}
\AtBeginDocument{\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
         \niib@log@debug{Converting endnotes to bibnotes}%
       \else
         \niib@log@inf{endnotes package not loaded\MessageBreak
           endnotes=false ignored}%
       \fi}}}
```

Options can now be processed.

```
\ProcessOptionsX[niib]<opt>
\AtEndOfPackage{
  \define@key[niib]{opt}{etex}
    {\niib@log@warn{Option 'etex' only available
       \MessageBreak when loading notes2bib}}}
\begingroup
  \@ifundefined{eTeXversion}
    {\ifniib@etex
       \niib@log@warn{e-TeX unavailable}%
       \global\niib@etexfalse
     \fi}
```

```
    {\ifniib@etex\else
       \niib@log@inf{Use of e-TeX disabled}%
     \fi}
\endgroup
```

7.5 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.

```
\newcounter{bibnote}
\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`.

```
\DeclareRobustCommand*{\bibnote}{%
  \stepcounter{bibnote}%
  \@ifnextchar[%]
    {\niib@bibnote}
    {\niib@bibnote[\thebibnote]}}
```

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

```
\DeclareRobustCommand*{\bibnotemark}{%
  \stepcounter{bibnote}%
  \@ifnextchar[%]
    {\niib@mark}
    {\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.

```
\DeclareRobustCommand*{\bibnotetext}[1][\thebibnote]{%
  \let\niib@next\niib@gobble
  \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`.

```
\AtBeginDocument{
  \@ifpackageloaded{biblatex}
    {\let\printbibnotes\printbibliography}
    {\DeclareRobustCommand*{\printbibnotes}
       {\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.

```
\DeclareRobustCommand*{\flushnotestack}{%
  \let\niib@taillist\niib@stack
```

```
  \ifniib@rerun\else
    \niib@checkrerun{tail}%
  \fi
  \ifx\@empty\niib@stack\@empty
    \niib@log@debug{Citation stack empty: nothing for
      \MessageBreak\noexpand\flushnotestack to do}%
  \else%
    \niib@log@debug{Flushing note citations to aux file}%
    \if@filesw
      \immediate\write\@auxout{%
        \string\niibtailcite{\niib@stack}}%
    \fi
    \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]}
```

7.6 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` 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\MessageBreak to 'tail'
    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{%
  \let\niib@next\niib@mark
  \niib@text{#1}{#2}}
```

`\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` 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@next` `\niib@gobble` To allow correct detection of following punctuation by `\niib@mark`, it has to follow `\niib@text`. However, to prevent category code issues, `\niib@text` can only take *one* argument. Hence, a "follow-on" macro is needed; this is provided here. The no-op here has to gobble a square-bracketed argument.

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

```
\def\niib@gobble[#1]{}
\let\niib@next\niib@gobble
```

\niib@text
\niib@out
\niib@stream

The `\niib@text` macro is the outer part of the writing system. It does not absorb the text of note, as without ε-TEX this is bad news.

```
\newcommand*{\niib@text}{%
  \@bsphack
  \@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
       \immediate\write\niib@out{\niib@msg}%
     \fi}
    {}%
  \if@filesw
    \expandafter\niib@write
  \fi}
```

\niib@write

The `\niib@write` macro deals with writing data verbatim to the BIBTEX file. When ε-TEX is available, this is easy; the macro absorbs the note text and uses `\unexpanded`.

```
\ifniib@etex
  \newcommand*{\niib@write}[2]{%
    \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}%
    \@esphack
    \niib@next[#1]}
```

Life is more awkward when ε-TEX is unavailable. The code here is based on a suggestion by Ulrich Diez, with alterations to the specifics needed here.

```
\else
  \newcommand*{\niib@write}[1]{%
    \begingroup
      \let\do\@makeother
      \dospecials
      \catcode`\{=1\relax
      \catcode`\}=2\relax
      \niib@write@{#1}}
\fi
```

\niib@write@

Two support macros are needed if ε-TEX is not available. `\niib@write@` works with the category codes altered as above, which means that text is written (essentially) verbatim to the BIBTEX file.

```
\newcommand\niib@write@[1]{%
```

```
  \long\def\niib@tempa##1{%
    \def\niib@tempa{##1}%
    \@onelevel@sanitize\niib@tempa\expandafter\endgroup
    \expandafter\def\expandafter\niib@tempa\expandafter{%
      \niib@tempa}%
    \niib@write@@{#1}}%
  \catcode`\^^M=10\relax
  \niib@tempa}
```

`\niib@write@@` Writing to the file is a very slightly altered version of the ε-TEX version. In this case, the debugging data is given here, after category code changes.

```
\newcommand*\niib@write@@[1]{%
  \niib@log@debug{Writing bibnote #1 contents
    \MessageBreak---\MessageBreak\niib@tempa\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{\niib@tempa\string}^^J%
    \string}^^J}%
  \@esphack
  \niib@next[#1]}
```

`\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` 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@tailcitelist` `\niibtailcite` To enable proper logging of citations when `sort=tail`, a similar system to the above is employed without the `\nocite` part.

```
\newcommand*{\niib@tailcitelist}{}
\newcommand*{\niibtailcite}[1]{%
```

```
  \edef\niib@tempa{#1}%
  \ifx\@empty\niib@tailcitelist\@empty
    \xdef\niib@tailcitelist{\niib@tempa}%
  \else
    \xdef\niib@tailcitelist{\niib@tailcitelist,\niib@tempa}%
  \fi}
```

`\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.

```
\AtBeginDocument{
  \@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}}}
```

7.7 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}[1]{%
```

```
  \niib@rerunfalse
  \expandafter\ifx\expandafter\@empty
    \csname niib@#1list\endcsname\@empty
    \expandafter\ifx\expandafter\@empty
      \csname niib@#1citelist\endcsname\@empty
      \niib@log@debug{No `#1' notes detected}%
    \else
      \niib@reruntrue
      \niib@log@debug{No `#1' notes found this run\MessageBreak
        but .aux files contained the `#1' requests:
        \MessageBreak\csname niib@#1citelist\endcsname}%
    \fi
  \else
    \expandafter\ifx\expandafter\@empty
      \csname niib@#1citelist\endcsname\@empty
      \niib@reruntrue
      \niib@log@debug{No `#1' requests in .aux file
        \MessageBreak but `#1' notes in this run:
        \MessageBreak\csname niib@#1list\endcsname}%
    \else
```

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

```
      \niib@checklists{#1}%
    \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 \MessageBreak
      `#1' 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` The business end of comparing the two lists. Two sweeps are made, to check that the lists match entirely.

```
\newcommand*{\niib@checklists}[1]{%
```

`\niib@list` `\niib@citelist` To allow `\niib@checklists` to handle both `sort=head` and `sort=tail` citations, the expanded list is needed for the `\@for` loops.

```
  \expandafter\edef\expandafter\niib@list\expandafter
    {\csname niib@#1list\endcsname}%
  \expandafter\edef\expandafter\niib@citelist\expandafter
    {\csname niib@#1citelist\endcsname}%
```

The loops can now being.

```
\@for\niib@tempa:=\niib@list\do{%
  \niib@reruntrue
  \@for\niib@tempb:=\niib@citelist\do{%
    \ifx\niib@tempa\niib@tempb
      \niib@rerunfalse
    \fi}
  \ifniib@rerun
    \niib@log@debug{Note \niib@tempa\space is a `#1' note
      \MessageBreak but request not in .aux file}%
  \fi}
\ifniib@rerun\else
  \@for\niib@tempa:=\niib@citelist\do{%
    \niib@reruntrue
    \@for\niib@tempb:=\niib@list\do{%
      \ifx\niib@tempa\niib@tempb
        \niib@rerunfalse
      \fi}
    \ifniib@rerun
      \niib@log@debug{Note \niib@tempa\space is set to `#1'
        in .aux\MessageBreak file but is not a `#1' note}%
    \fi}
\fi}
```

`\niib@taillist` 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{%
  \niib@rerunfalse
  \niib@checkrerun{head}%
  \flushnotestack
  \@ifundefined{niib@out}{}
    {\immediate\closeout\niib@out
     \niib@log@debug{Closed BibTeX database file\MessageBreak
       \niib@stream}}}
```

8 Notes

[1] Note for the first example.

[2] Note for the second example.

[3] Note for the third example.

[4] Some `\verb`-like output.

9 Change History

v1.0

General: Initial public release 1

v1.0a

`\citenote`: New macro 13

10 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.

O

options:

P

R

S

T