The Law module for the Camel Bibliography Engine*

Frank G. Bennett, Jr.†

July 12, 1995

Abstract

The `Law` module for the Camel bibliography package and BibTeX attempts to implement fully automated typesetting of citations in the so-called Blue Book style used in the publication of legal materials. This demanding style requires context sensitive in-footnote cross-referencing between citations.

An adage in office management is that you should only touch incoming paper once; to respond to it, to file it, to forward it, or to destroy it. A number of commercial citation database managers provide a facility for "filing" citations in a flexible form, the idea being to extend this principle to citations as well as paper. A hanging point for this strategy as been in-text context-sensitive citation styles. Database managers are at their best in exporting entire bibliographies and lists of authorities. Some packages are capable of scanning a document for citation "tags", which eliminates the need to separately select bibliography items in the database. Some, too, can replace "tags" in the document with the formatted text of a citation. But once the text is replaced, the format of citations added in this way is fixed; conversion is a one-way process.

A more serious problem is that, while the database manager can easily identify the *tag*, it is far more difficult, without logical markup, to identify its *context*—whether it occurs in a footnote, how many items were in the preceding footnote, how many articles by the same author are cited in the document, and so forth. As a result, citation formatting of cross-referenced styles is still generally done by hand.

One of the most demanding cross-referenced styles is that laid down in *A Uniform System of Citation*, or "the Blue Book", for the citation of legal materials.

*This file has version number 0.1a dated 1995/07/08. The documentation was last revised on 1994/12/07.

†Lecturer in the Commercial Laws of the Far East, School of Oriental and African Studies, University of London. Acknowledgements of the numerous people who have provided comments and suggestions in the development of this module and of the Camel package itself are listed in the users' guide to that software.

This style has survived the era of computerization largely because most U.S. law journals using the style are edited by highly competitive law students. Staff members contribute their editorial time to their journal free of charge, because of the value of listing law journal membership on their resume. The Blue Book style minimizes the bulk of citation text, while conveying sufficient information to the reader for the location of cited material. It is also designed to provide all information required for the location of *primary* legal authority from the face of any citation to it, without tracing down cross-references.

A first attempt to address this problem was made in the complementary LexiTeX and LexiBib packages, for LaTeX and BibTeX respectively. Lessons learned from developing these packages suggested that a comprehensive formatting package for legal citations could serve as the foundation for a modular, highly generalized citation and bibliography formatting system. The logical code from these packages was excised and used in the drafting of the Camel bibliography engine. The typesetting code was then used to assemble the Law module for Camel.

The implications of such a system are particularly interesting if it comes to be widely used in the publication of law journals or, even better, in the publication of court judgements. The efficiency with which citations can be reported to citation services would be increased, since BibTeX-format database entries are in a standard format that can be processed electronically. These same lists could be made available as text-searchable databases of authority. Network discussions of legal issues could be accompanied by growing lists of annotated authority, available for all to use.

To use this package, all you should need to do is unpack the files by running `law.ins` (thanks to Robin Fairbairns for pushing me toward a standard-ish method of installation!), copy the style files to their respective homes, and follow the markup conventions described in the Camel manual. The same should apply to any other bibliography modules produced for Camel in the future, no matter what sort of contortions they put your citation data through before they end up on the page. If it has extensive documentation, it's *not* a Camel module!

You are not under an obligation to enjoy this package. If you have complaints, you can contact me on `fb@soas.ac.uk`.

1 The style code

1.0.1 Entry type specific functions

The functions below should not be used as general utilities; they are designed specifically for use with a particular entry type. While they could be placed directly into the entry functions to which they apply, defining them separately helps improve the transparency of the code.

```
⟨*bstheader⟩
"     This is the 'law' style for BibTeX and Camel"
⟨/bstheader⟩
⟨*bstfunctions⟩
```

`build.bridges` I'm not actually sure whether this function belongs in `camel.dtx` or in `law.dtx`. What it does is to insert (or not insert) an appropriate `\bridges` declaration into the `.bbl` file for the current citation. It is governed by a string existing on the stack when it is invoked, and leaves nothing on the stack. Toggles are `schedules`, `sections` and `articles`. Season to taste.

```
FUNCTION {build.bridges}
 { duplicate$ empty$
   { pop$ skip$ }
   { duplicate$ "sections" =
     { pop$ "\bridges{\ \S~}{}{\ \S\S~}{\ }{\ \S~}{\ \S\S~}"
     "" "" must.must.must
     }

     { duplicate$ "articles" =
       { pop$
         "\bridges{\ art.~}{}{\ arts.~}{\ }{\ art.~}{\ arts.~}"
         "" "" must.must.must
       }
       { "schedules" =
         { "\bridges{\ sched.~}{}{\ scheds.~}{\ }{\ sched.~}{\ scheds.~}"
           "" "" must.must.must
         }
         'skip$
         if$
       }if$
     }if$
   }if$
}
```

`get.a.kinda.sort.key` It is early days for the sorting of bibliographies in Camel. This will give a rough key.

```
FUNCTION { get.a.kinda.sort.key }
{     author empty$
     { title "*" "forward" gather.chars pop$
       duplicate$ "l" change.letter.case "the" =
       { pop$ "*" "forward" gather.chars pop$ swap$ pop$ }
       { duplicate$ "l" change.letter.case "a" =
         { pop$ "*" "forward" gather.chars pop$ swap$ pop$ }
         { swap$ pop$
         }if$
       }if$
     }
     { author #1 "{ll}" format.name$
     }if$
}
```

`j.format.division` This function assembles the various fields relevant to the division of a Japanese court into a single string.

```
FUNCTION {j.format.division}
 { division empty$
```

```
  { "" }
  { dc.. court "end" first.in.second
    { pop$ " No.~" 's := "endlabel" 't := }
    { pop$ sc.. court "start" first.in.second
      { pop$ "No.\ " 's := "frontlabel" 't := }
      {  pop$ " " 's :=  "endlabel" 't :=
      }if$
    }if$
  }if$
  division s divno t field.tag.no.combine
}
```

1.0.2 Entry type functions

Now we define the type functions for all entry types that may appear in the `*.bib` file—e.g., functions like '`article`' and '`book`'. These are the routines that actually generate the `*.bbl` file output for the entry. These must all precede the `READ` command. In addition, the style designer should have a function '`default.type`' for unknown types.[1]

`article` This function performs the necessary operations for exporting a valid LaTeX citation to an article. For this and for all citation types defined in the LexiBib style, the goal is to provide reasonably complete commentary, so that anyone wanting to alter the behaviour of the style can set to work with a fair degree of confidence about what needs to be done to achieve a particular result.

```
FUNCTION {article}
```

The `article` entry is used for all kinds of material, so it ends up as one of the most complex entries. Before we do anything, we have to check whether the default Camel bridges are acceptable. There are two situations to watch out for. First, if the volume and the number are both non-empty, we need to add a special set of substitute bridges.

```
{ volume empty$ not number empty$ not and
  volume empty$ number empty$ year "mo.dd.yy" format.date
  pop$ itemcount #1 = not and and or
  { "\bridges{,\ p.~}{,\ }{,\ }{,\ }{\ at~}{\ at~}"
    "" "" must.must.must
    newline$
  }
```

Second, the `@article` entry type, like many entry types in the Blue Book style normally places white space between the title and a pinpoint page number. If the title ends in a numeral this will be confusing, so the Blue Book requires that we separate the two with a comma in this case. The following adjustment to bridging punctuation accomplishes this purpose.

```
  { title type.last.char "numeral" =
    { "\bridges{,\ }{,\ }{,\ }{\ }{\ at~}{\ at~}"
      "" "" must.must.must
```

[1] This comment by Oren Patashnik.

```
      newline$
    }
    'skip$
    if$
  }if$
```

Then we write the citation leader, to prepare for outputting the actual content of the citation text.

```
 "\lexibib{article}{" cite$ "}{" must.must.must
 get.a.kinda.sort.key
 "}{" "" must.must.must
```

The author name is pushed to the stack, followed by a toggle to trigger last-name-only formatting. Then the `format.names` reduces this to a single, appropriately-formatted string, possibly the null string. All that is left to do is push a set a braces, a null string to make up three arguments to the export routine, and write the lot on the output file unconditionally, using `must.must.must`.

```
author  "lastonly" format.names
"}{"
""
must.must.must
```

Next comes the title of the article. This is not specially formatted; we simply push the title field, then a warning string followed by a check for whether it is empty or not, then braces and a null string, and write again.

```
title "title" check
"}{"
""
must.must.must
```

The next bit is actually rather thorny. There are three possible cases. The first is where both a volume number and an issue number exist. In this case, we use a verbose form of reference. In the second, there is no volume number, but possibly an issue number. This is the proper form for Commonwealth legal materials, and requires the year in brackets, followed by the issue number to show the volume within the year. Third, we may simply have a volume number by itself. This calls for the citation form for most journals, and U.S. case reporters.

```
 volume empty$ not number empty$ not and
 { journal "journal" check ", v.~" volume must.must.must
   ", n.~" number "}{" must.must.must }
 { volume empty$ not
   { volume "\ " journal "journal" check empty.to.null
     might.ifone.must
     "}{" "" "" must.must.must}
   { year "mo.dd.yy" format.date itemcount #1 =
     { "[" swap$ "] " iftwo.might.iftwo
     number "\ " journal "journal" check might.ifone.must
     "}{" "" "" must.must.must}
     { pop$ number empty$
       { "\\" journal "journal" check "\\" must.must.must
```

```
          "}{" "" "" must.must.must }
        { number "\ " journal "journal" check might.ifone.must
          "}{" "" "" must.must.must
        }if$
      }if$
    }if$
  }if$
```

The Blue Book does not like page ranges so we need to clean out anything following a dash in the `pages` field. The `short` option toggles this behaviour on. We also check to see that the page is not empty. This is followed by braces, a null string, and output.

```
pages "short" format.pages "pages" check
"}{"
""
must.must.must
```

We add the year next, but only if the `volume` field is non-empty (if `volume` is empty, we'll have put the year is as a bracketed volume number, Commonwealth-style. We could use just the year (this is normal for Blue Book style, but we'll add the month for good measure, if it's been provided in the `year` field.

```
volume empty$ not
  { "(" year "mo.dd.yy" format.date ")" must.must.must }
  { year "mo.dd.yy" format.date itemcount #1 =
    'pop$
    { "" "" must.must.must
    }if$
  }if$
```

Finally, we have to tangle with cross-references. Yuck. (Not a complaint about BibTeX, just a general response to the design problems inherent in the task). Formatting depends on whether there is a `crossref` entry. Hear that, guys? If you don't use the `crossref` field, we'll short-change you on formatting service.

```
crossref empty$
```

If there is no `crossref`, we just push a brace and a couple of nulls, and write. Done! Hurray!

```
{ "}" "" "" must.must.must }
```

But if there *was* a `crossref`, we've got work to do. Darn. The first thing we do is have a look at `booktitle`. This should be non-null in this situation; there's no sense setting up a cross-reference to an individual volume of a journal unless there's something special to be said about it.

```
{ booktitle empty$
```

So if no `booktitle` is found, we whinge and format as for a no-cross-reference entry.

```
  { "no booktitle (name of special issue) for "
     cite$ "/" crossref ".  Why a crossref?" * * * * warning$
    "}" "" "" must.must.must }
```

If there *is* a `booktitle`, though, and if the `volume` field is non-empty (which means that we just printed, or at least should have printed, the year), we close the parens following the year (which is opened by LexiTeX), and open another (which will be closed by LexiTeX). That's it for conditional punctuation; we follow with `booktitle`, which should be the title or subject description of the special issue of the journal.

```
  { volume empty$ {""} {") ("} if$ booktitle "}" must.must.must}
  if$
}if$
```

Add a fresh new line in the export file, and we're done! Whew!

```
newline$
}
```

`book` This is the entry for books, which includes individual volumes in a series, and multi-volume works with a single title. Correct me if I'm wrong, but I think this latter citation type is not supported by BibTeX. LexiBib manages it by allowing the user to specify the volume number in the text using the optional `:<number>:` argument to the `\lexicite` tag.

```
FUNCTION {book}
```

This entry type normally places white space between the title and a pinpoint page number. If the title ends in a numeral this will be confusing, so the Blue Book requires that we separate the two with a comma in this case. The following adjustment to bridging punctuation accomplishes this purpose.

```
{ title type.last.char "numeral" =
    { "\bridges{,\ }{}{,\ }{\ }{\ at~}{\ at~}"
      "" "" must.must.must
      newline$
    }
    'skip$
    if$
```

Next, after the opener, we push the opening macro tag for a book entry, the nickname of the citation, and a couple of braces. This is all mandatory and can safely be given unconditional export.

```
  "\lexibib{book}{" cite$ "}{" must.must.must
   get.a.kinda.sort.key "" "}{" must.must.must
```

A non-empty `volume` field means we need a leading volume number in the Blue Book style. If a volume number (or anything else) is found in the `volume` field of a book entry, we replace it with a `volno` macro. This will expand in the document to whatever the author has specified using the optional `:<number>` argument to `\lexicite`. For example, volume 8 of `holdsworth` would be: `\lexicite:8:{holdsworth}`.

```
  volume empty$
  { "" }
  { "\volno " }
  if$
```

The author comes first. We push the contents of the `author` field, then the toggle string "`firstinitial`", and run the `format.names` function to produce the name formatted properly for a book entry. Then we push fa couple of braces and force all three items (`\volno` or null, author, and braces) onto the output.

```
138  author
139  "firstinitial" format.names "}{" must.must.must
```

The title is very straightforward. We push the title, then check to be sure it's non-empty, then a couple of braces go up, filled out with a null string, and force all three onto the output.

```
140  title "title" check "}{}{}{" "(" must.must.must
```

We're now in the final "field" of the LaTeX entry. This is mainly for the year, but we also give the name of the editor(s) or translator(s) if present. If there is no editor or translator, we'll put a series name here, to help identify the source. We don't put both, because this would confuse things (there can be book editors and series editors too, and the Blue Book style is too streamlined to distinguish the two elegantly). So our first task is to see if there is an editor...

```
141    editor empty$ translator empty$ and
```

... and if there is none, we put in a series name if it exists. The `series` BibTeX field should be used for the name of the series of which a volume forms a part, but some folks might accidentally use `booktitle`. We'll be forgiving and accept it anyway.

```
142    { series empty$ booktitle empty$ and
```

If both `series` and `booktitle` are empty, we've nothing to do.

```
143      'skip$
```

If at least one is non-empty, we push both, and cull them to just one using the `either.or` function. If only one is non-empty, this function leaves that one; if both are non-empty, the `either.or` function whinges and chooses one arbitrarily.

Next, we push a bridge and a series item number, and a toggle for the `field.tag.no.combine` function. The `endlabel` toggle causes this function to put the bridge and the number after the series name, if a number exists, and push the lot back as a single item on the stack. Otherwise it leaves just the series name.

And last, we put up a comma to close, and do a mandatory export of the lot.

```
144      { series booktitle either.or " No.~" number
145      "endlabel" field.tag.no.combine ", " "" must.must.must }
146      if$
147    }
```

If either the editor or the translator fields were not empty, we format the editor or translator name instead, and put those details here.

```
148    { editor translator either.or "firstinitial" format.names
```

We need to append the correct designator, either "ed." or "trans.". The `either.or` function will use the second item pushed if both are non-empty, so we take advantage of this "feature" in making our choice of designators; the `ed.` or `eds.` strings

are only used if the `translator` field is empty. And finally, we push a null string to round out, and do a compulsory export.

```
      translator empty$
      { editor num.names$ #1 >
        { " eds.\ " }
        { " ed.\ " }
        if$
      }
      { " trans.\ " }
      if$
      "" must.must.must
    }if$
```

We also need to indicate the edition, if any.

```
  edition " ed.\ " "" might.ifone.must
```

The year itself is easy. We push the year, do a check to issue a warning if necessary, then run format.date over it, which yields the year in `theyear`, which can be pushed back onto the stack. Then we fill out to six LaTeX fields in all, and do a compulsory export.

```
  year "mo.dd.yy" format.date ")}" "" must.must.must
```

A new line for a new macro, and we're done! Rejoice! On to the next function definition!

```
  newline$
}
```

incollection

This is for those nasty entries that are created when someone publishes an article in a collection of essays edited by someone else.

```
FUNCTION{incollection}
{ "\lexibib{incollection}{" cite$ "}{" must.must.must
   get.a.kinda.sort.key "" "}{" must.must.must
  author "lastonly" format.names "author" check
  "}{" "" must.must.must
  title "title" check
  "}{" "" must.must.must
  chapter empty$
    { "" }
    { "\\" type empty$
      { "chapter " chapter " of \\" * * * }
      { type " " chapter " of \\" * * * *
      }if$
    }if$
  booktitle "booktitle" check
  "}{" must.must.must
  pages "short" format.pages "pages" check
  "}{" "(" must.must.must
% We're now in the final ``field'' of the \LexiTeX{} entry.
% The coding here is the same as for a "book" entry; the
% reader is referred to that entry for the commentary
% on the following code.
```

```
%    \begin{macrocode}
    editor empty$ translator empty$ and
    { series empty$
      'skip$
      { series " No.~" number
      "endlabel" field.tag.no.combine ", " "" must.must.must }
      if$
    }
    { editor booktranslator either.or "firstinitial" format.names
      booktranslator empty$
      { editor num.names$ #1 >
        { " eds.\ " }
        { " ed.\ " }
        if$
      }
      { " trans.\ " }
      if$
      "" must.must.must
    }if$
  edition " ed.\ " "" might.ifone.must
  year "mo.dd.yy" format.date ")}" "" must.must.must
  newline$
}
FUNCTION{inbook}
{ "\lexibib{inbook}{" cite$ "}{" must.must.must
  get.a.kinda.sort.key "" "}{" must.must.must
  author "firstinitial" format.names "author" check
  "}{" "" must.must.must
  title "title" check
  "}{" "" must.must.must
  chapter empty$
    { "\\in \\" }
    { type empty$
      { "\\chapter " Chapter " of \\" * * }
      { "\\" type " " Chapter " of \\" * * * *
      }if$
    }if$
  booktitle "booktitle" check
  "}{" must.must.must
  pages "short" format.pages "pages" check
  "}{" "(" must.must.must
% We're now in the final ``field'' of the \LexiTeX{} entry.
% The coding here is the same as for a "book" entry; the
% reader is referred to that entry for the commentary
% on the following code.
%    \begin{macrocode}
    translator empty$
    { series empty$
      'skip$
      { series " No.~" number
```

```
      "endlabel" field.tag.no.combine ", " "" must.must.must }
      if$
    }
    { translator "firstinitial" format.names
      " trans.\ "
      "" must.must.must
    }if$
  edition " ed.\ " "" might.ifone.must
  year "mo.dd.yy" format.date ")" "}" must.must.must
  newline$
}
FUNCTION{booklet}
{ "\lexibib{booklet}{" cite$ "}{" must.must.must
  get.a.kinda.sort.key "" "}{" must.must.must
  author "full" format.names "}{" "" must.must.must
  "\\" title "\\}{}{}{(" must.must.must
  howpublished ", " "" might.ifone.must
  year "mo.dd.yy" format.date
  ")}" "" must.must.must
  newline$
}

FUNCTION {techreport}
{"\lexibib{techreport}{" cite$ "}{" must.must.must
  get.a.kinda.sort.key "" "}{" must.must.must
  institution author either.or.nowarning
  "full" format.names "author & institution" check
  "}{" title "title" check must.must.must
  "}{}{}{" "(" "" must.must.must
  author empty$
    'skip$
    {institution "\ "  "" might.ifone.must
    }if$
  type empty$
    { "Technical report" }
    { type
    }if$
  type empty.to.null "Cmnd" =
  { "\ " }
  { " No.~"
  }if$
  number "endlabel" field.tag.no.combine
  ", " "" must.must.must
  year "mo.dd.yy" format.date
  "" ")}" must.must.must
  newline$
  }

FUNCTION {mastersthesis}
{"\lexibib{mastersthesis}{" cite$ "}{" must.must.must
```

```
  get.a.kinda.sort.key "}{" "" must.must.must
 author
 "full" format.names "author" check
 "}{" title "title" check must.must.must
 "}{}{}{("
 type empty$
   { "Master's Thesis" }
   { type
   }if$
 ", " must.must.must
 institution "institution" check ", " "" might.ifone.must
 year "mo.dd.yy" format.date
 ")" "}" must.must.must
 newline$
 }
```

Cases Law cases are all entered using the `@CASE` entry type. The formatting of citations varies from jurisdiction to jurisdiction, so the behaviour of citations of this type is controlled via a `jurisdiction` field. Supported jurisdictions are listed in the user guide. Below, the functions for each jurisdiction are defined first, followed by the `case` function itself.

```
FUNCTION {case}
```

Like the `article` entry, the `case` entry must make some pretty fine decisions about how to format material fed to it. As a consequence, it is complex, but similar to the `article` entry in many particulars.

```
{ cites empty.to.null "=" *
 journal empty$
 { parse.one.cite }
 { volume empty.to.null 'volume.var :=
   number empty.to.null 'number.var :=
   journal 'journal.var :=
   pages empty.to.null 'pages.var :=
   year empty.to.null 'year.var :=
 }if$
 volume.var empty$ not number.var empty$ not and
 { "\bridges{,\ p.~}{,\ }{,\ }{\ }{\ at~}{\ at~}"
   "" "" must.must.must
   newline$ }
 'skip$
 if$
"\lexibib{case}{" cite$ "}{" must.must.must
 get.a.kinda.sort.key "" "}{}{" must.must.must
 title empty$
 {  "Decision of the " court "court" check "" must.must.must
     " (" j.format.division ")" iftwo.might.iftwo
     ", " "" "" must.must.must
     casedate "month.dd.yy" must.must.must
     "}{" "" "" must.must.must
```

```
  }
  { title "}{" "" must.must.must }
  if$
  volume.var empty$ not number.var empty$ not and
  { journal.var "journal" check ", v.~" volume.var must.must.must
    ", n.~" number.var "}{" must.must.must }
  { volume.var empty$
    { "[" year.var "mo.dd.yy" format.date "] " iftwo.might.iftwo
      number.var "\ " journal.var "journal" check might.ifone.must
      "}{" "" "" must.must.must }
    { volume.var "\ " journal.var "journal" check empty.to.null
      might.ifone.must
      "}{" "" "" must.must.must
    }if$
  }if$
pages.var "short" format.pages "pages" check
"}{"
"("
must.must.must
volume.var empty$
  'skip$
  { year.var "mo.dd.yy" format.date "" "" must.must.must
  }if$
crossref empty$
{ ")}" "" "" must.must.must }
{ booktitle empty$
  { "no booktitle (name of special issue) for "
     cite$ "/" crossref * * * warning$
    ")}" "" "" must.must.must }
  { volume.var empty$ {""} {") ("} if$ booktitle "}" must.must.must}
  if$
}if$
{ duplicate$ "=" = not }
{ parse.one.cite
  "={" "" "" must.must.must
  volume.var empty$ not number.var empty$ not and
  { journal.var "journal" check ", v.~" volume.var must.must.must
    ", n.~" number.var "}{" must.must.must }
  { volume.var empty$
    { "[" year.var "mo.dd.yy" format.date "] " iftwo.might.iftwo
      number.var "\ " journal.var "journal" check might.ifone.must
      "}{" "" "" must.must.must }
    { volume.var "\ " journal.var "journal" check empty.to.null
      might.ifone.must
      "}{" "" "" must.must.must
    }if$
  }if$
pages.var "short" format.pages "pages" check
"}{"
""
```

```
must.must.must
volume.var empty$
  'skip$
  { "(" year.var "mo.dd.yy" format.date ")" must.must.must
  }if$
crossref empty$
{ "}" "" "" must.must.must }
{ booktitle empty$
  { "no booktitle (name of special issue) for "
     cite$ "/" crossref * * * warning$
    "}" "" "" must.must.must }
  { volume.var empty$ {""} {") ("} if$ booktitle "}" must.must.must}
  if$
}if$
}while$
pop$
newline$
annote empty.to.null write$ newline$
}
```

The following item adds annotations; this may be eliminated by stripping with `noannotes`.

```
⟨*!noannotes⟩
% annote empty.to.null write$ newline$
⟨/!noannotes⟩
% }
```

`j.statute` This function applies to Japanese statutory materials.

```
FUNCTION {j.statute}
 { "\lexibib{jstatute}{" cite$ "}{" must.must.must
   get.a.kinda.sort.key "" "}{}{" must.must.must
   title "title" check "}{}{}{" "" must.must.must
   "Law no.~" number "number" check " of "
   iftwo.might.iftwo
   year "yy" format.date "}" "" must.must.must
   newline$
 }
```

`s.statute` This function formats a statute entry for Singapore.

```
FUNCTION { s.statute }
{ "\lexibib{statute}{" cite$ "}{" must.must.must
   get.a.kinda.sort.key "" "}{}{" must.must.must
  title "title" check ", No.~" number
  "endlabel" field.tag.no.combine
  number empty$
    { "\ " * }
    { "\ of " * }
    if$
  year "mo.dd.yy" format.date "}{}{}{}" must.must.must
  newline$
}
```

```
FUNCTION { e.statute }
{ "\lexibib{statute}{" cite$ "}{" must.must.must
   get.a.kinda.sort.key "" "}{}{" must.must.must
  title "title" check "\ "
  year "mo.dd.yy" format.date must.must.must
  "}{}{}{}" "" "" must.must.must
  newline$
}
```

`statute` This function selects the correct statute entry function.

```
FUNCTION { statute }
 { type build.bridges
   jurisdiction empty.to.null duplicate$
   "japan" =
   { pop$ j.statute }
   { duplicate$ "singapore" =
     { pop$ s.statute }
     { duplicate$ "england" =
       { pop$ e.statute }
       { pop$ "IMPORTANT: unknown jurisdiction for " cite$ * warning$
       }if$
     }if$
   }if$
 }
```

`default.type` We use the `book` type as our default type. When `manual` is completed, we should probably use that type instead.

```
FUNCTION {default.type} { book }
```

⟨/bstfunctions⟩

1.1 Macro definitions

We don't define any macros for abbreviating law journal names. Instead, we will use Blue Book abbreviations "native", with a special syntax (probably the full form in syntax: "`\gobble{Accountant}{}`" immediately after the abbreviation) for resolving ambiguous abbreviations. Meanwhile, trust me: use the Blue Book abbreviations and take this upcoming facility on faith. And besides, do you ever *need* to spell out journal and reporter names?

1.2 Execution

With all preliminaries out of the way, our first act is to read in the entries from `*.bib`..

⟨*bstfunctions⟩

```
READ
```

Then we say "Hi" to the user. It would be nice to make this the first message, but the structure of BibTeX style files dictates that it will follow any warnings about missing entries.

```
EXECUTE {hello}
```
⟨/bstfunctions⟩

2 Camel style code

⟨*lawcitestyle⟩
```
\ProvidesFile{law.cst}[1995/01/08]
```

2.1 Word list

A list of inter-words and their corresponding expansions must be provided.

```
{\catcode`\_=13%
 \catcode`\^=13%
\gdef\@law@wordlist{%
  \\{\item}{\item}%
  \\{and}{_\ ^And }%
  \\{but-see}{_\ ^But see }%
  \\{,}{_; }%
  \\{;}{_; }%
  \\{:}{_; }%
  \\{eg}{_; ^E.g.~}%
  \\{accord}{_; ^Accord }%
  \\{see}{_; ^See }%
  \\{see-also}{_; ^See also }%
  \\{cf}{_; ^Cf.~}%
  \\{compare}{_; ^Compare }%
  \\{with}{ with }%
  \\{contra}{_; ^Contra }%
  \\{but-cf}{_\ ^But cf.~}%
  \\{see-generally}{_. See generally }%
  \\{affirmed}{_, ^aff'd }%
  \\{reprinted-in}{_; ^Reprinted in }%
  \@law@nomatch}
}
```

2.2 Print routines

Every `.cst` file must define a `\@law@print` macro. This is the macro that prints the citation, both in the text of the document and in the bibliography. The toggles set by the CAMEL engine allow a high degree of refinement in the formatting of citations. For the styles currently used in publishing with LaTeX, relatively little of this power is required. In-document citation styles are much more demanding; if you digest the operation the following `\@law@print` macro, which is suitable for formatting legal citations, you will find the drafting of author-date styles and the like quite straightforward by comparison.

```
\gdef\@law@print{%
```

In this particular style, a long citation (for the bibliography, for example) and a first in-text citation are identical. These forms have separate toggles, so we start by equating them.

```
\if@law@firstuseofcite\@law@longcitetrue\fi%
```

This toggles the printing on and off. This toggle is set by the `n` option to the `\source` command.

```
\if@law@printcite%
 \begingroup%
 \def\@law@firstslash{\begingroup\def\\{\@law@secondslash}%
  \the\ltokspecialface}%
 \def\@law@secondslash{\endgroup\def\\{\@law@firstslash}}%
 \def\\{\@law@firstslash}%
```

There are two halves to the macro; one for long cite forms, the other for short. Long cite forms are in the first half.

```
\if@law@longcite%
```

Long citations are pretty straightforward; we've gathered all the information we needed; now we just need to plunk it all down in order, pretty much. The one exception is the page bridges. If the location page exists, we leave the singular bridge alone. If no location page exists, *and* the pinpoint reference is plural, then we install a plural bridge.

```
\ifcat$\the\@ltok@citepage$%
 \if@law@multipages%
  \global\@ltok@conetop\@ltok@conetopplural%
 \fi%
\fi%
```

First to print is the author field (the second argument to the citation declaration command), followed by the author-to-title punctuation bridge. The enclosing braces limit the scope of the special active character definitions of `^`, `_` and `|`.

```
\global\ltokspecialface=\@ltok@authoroptionface%
{\the\@ltok@authormainface%
\@law@barinfull\the\@ltok@author}\the\@ltok@atot%
```

Next comes similar treatment for the title.

```
\global\ltokspecialface=\@ltok@titleoptionface%
{\the\@ltok@titlemainface%
\@law@barinfull\the\@ltok@name}\the\@ltok@ttocone%
 \@law@longrecurse%
```

This else marks the boundary between long-form citations (which we have seen are relatively simple to print) and short-form citations (which are rather complex). This `\else` matches the `\if@law@longcite` conditional.

```
\else%
```

The footnote number and the accompanying bridge should not appear in a short-form citation if the citation being printed first occurred in the current footnote. Rather than suppressing printing, we just set the relevant tokens to nil.

```
  \ifnum\the\@ltok@pageorfootno=\the\c@law@footnote\relax%
    \xdef\@law@temp{\the\@ltok@whereitsat}%
    \xdef\@law@temptwo{\the\@ltok@infoot}%
    \ifx\@law@temp\@law@temptwo%
     \global\@ltok@whereitsat{}%
     \global\@ltok@pageorfootno{}%
    \fi%
  \fi%
```

Special forms of tidying-up may be appropriate to each of the four classes of citation. An appropriate cleaning macro is called here.

```
\csname @law@\the\@ltok@citetype preformat\endcsname%
```

If it has been found that the same citation has been used immediately previous to this instance, we use *Id.* If the `\@justabove` test showed that *Id.* by itself is appropriate, we prepare to eat any period immediately following the citation macro. Otherwise, we tack on the pinpoint reference and its accompanying bridge. A fail-safe conditional reverts to the original plan if the pinpoint reference is empty.

```
  \if@justabove%
   {\def\,{,}%
    \Id%
     \if@l@quiteexact%
      \gdef\@law@gobble{\@ifnextchar.{\@gobble}{}}%
     \else%
      \ifcat$\the\@ltok@argtwo$%
\message{Did define!}%
        \gdef\@law@gobble{\@ifnextchar.{\@gobble}{}}%
      \else%
        \the\@ltok@atbridge{\@law@barkill\the\@ltok@argtwo\relax}%
        \@law@fetchparas%
        \@law@shiftparas%
        \@law@shortrecurse%
      \fi%
     \fi}%
```

If *Id.* was not called for, we have to create a proper short-form reference. This `\else` corresponds with the `\@justabove` toggle, above.

```
  \else%
```

The author will be used in any case. Note that the author and its following bridge may have been set to nil, above.

```
   \global\ltokspecialface=\@ltok@authoroptionface%
   {\the\@ltok@authormainface%
%  print whatever there if for the author
   \@law@barinshort\the\@ltok@author}\the\@ltok@atot%
```

The title information is also used, if it is present.

```
   \global\ltokspecialface=\@ltok@titleoptionface%
   {\the\@ltok@titlemainface%
```

If we have specified a nickname for this cite for 'hereinafter' references, we use it instead of the title of the work. ¡*parabeta¿ [DWEEZLE: a test and a token register replacement goes here.] ¡/parabeta¿

```
\@law@barinshort\the\@ltok@name}%
```

At this point, we must make decisions concerning whether to use the *supra* cross-referencing form. This depends on the type of citation we are working on. The `\@nosupra` condition is set to true if we are working on a case or a statute, otherwise it is set to false.

```
\xdef\@law@temp{\the\@ltok@citetype}%
 \ifx\@law@temp\@law@case%
  \@nosupratrue%
 \else%
   \ifx\@law@temp\@law@statute%
    \@nosupratrue%
   \else%
    \@nosuprafalse%
   \fi%
 \fi%
```

Now we put the result of the above test to work. First, the citation form for cases or statutes, which do not permit the *supra* form.

```
 \if@nosupra%
  \the\@ltok@ttocone%
  \@law@shortprint%
  \@law@shortrecurse%
 \else%
  \supra\the\@ltok@whereitsat\the\@ltok@pageorfootno%
  \the\@ltok@atbridge%
  {\def\,{,}%
  {\@law@barkill\the\@ltok@argtwo\relax}}%
 \fi%
```

In order, these close `\if@justabove`, `\if@law@longcite` and `\if@law@printcite`.

```
  \xdef\@law@temp{\the\@ltok@citetype}%
   \ifx\@law@temp\@law@statute%
    \ifcat$\the\@ltok@citefirst$%
      \the\@ltok@ptoctwo{\@law@barinshort\the\@ltok@citelast}%
    \fi%
   \fi%
 \fi%
\fi%
\endgroup%
\fi%  <- end of if@law@printcite
```

The `\@law@gobble` here will consume a period inserted automatically after a forced footnote, if the printed form turned out to be an *Id.*. See the concluding code of `\@law@setup`.

```
\global\@law@longcitefalse\global\def\volno{}}%
```

\@law@longrecurse This macro prints the tail end of long citations until parallels are exhausted.

```
\def\@law@longrecurse{%
  \@law@longprint%
  \ifnum\the\c@law@parapin>0\relax%
    \loop%
      \addtocounter{law@paracounter}{1}%
    \ifnum\the\c@law@paracounter<\the\c@law@parapin\relax%
      \@law@pincut\@ltok@argtwo\frompinlist%
      ; \@law@longprint%
    \repeat%
  \else%
    \loop%
      \addtocounter{law@paracounter}{1}%
    \ifnum\the\c@law@paracounter>\the\c@law@paranormal\relax%
    \else%
      ; \@law@longprint%
    \repeat%
  \fi}
\def\@law@shortrecurse{%
  \ifnum\the\c@law@parapin>0\relax%
    \loop%
      \addtocounter{law@paracounter}{1}%
    \ifnum\the\c@law@paracounter<\the\c@law@parapin\relax%
      \@law@pincut\@ltok@argtwo\frompinlist%
      ; \@law@shortprint%
    \repeat%
  \else%
    \loop%
      \addtocounter{law@paracounter}{1}%
    \ifnum\the\c@law@paracounter>\the\c@law@paranormal\relax%
    \else%
      ; \@law@shortprint%
    \repeat%
  \fi}
```

\@law@longprint If CAMEL sees more than one pinpoint page or section, the bridges preceding the page references must be set to their plural form. This change applies to both long and short form citations. It has to be cloned in each, however, because the decision has to be taken with respect to each cite in a string of parallels.

```
\def\@law@longprint{%
  \begingroup%
  \@law@fetchparas%
  \@law@tidybridges%
  \if@law@multipages%
    \@ltok@conetop\@ltok@conetopplural%
    \@ltok@atbridge\@ltok@atbridgeplural%
  \fi%
```

If for some reason either the pinpoint argument is nil, or it *and* the location page token register is nil, we set the accompanying bridges to nil.

```
  \ifcat$\the\@ltok@argtwo$%
   \global\@ltok@ptop{}%
   \global\@ltok@atbridge{}%
   \ifcat$\the\@ltok@citepage$%
     \global\@ltok@conetop{}%
    \fi%
   \fi%
 \global\ltokspecialface=\@ltok@citefirstoptionface%
 {\the\@ltok@citefirstmainface%
 \@law@barinfull\the\@ltok@citefirst}\the\@ltok@conetop%
```

Next comes the location page and its following bridge. Both of these may be blank. No funny business with the shorthand active characters is required, since we assume this will not contain any special text for which they will be required.

```
 \the\@ltok@citepage\the\@ltok@ptop%
```

A special use of `\,` is defined before we expand the optional argument stuff.

```
  {\def\,{,}\@law@barkill%
   \expandafter\the\@ltok@argtwo%
   \relax}%
```

And finally, we print the final portion of the citation.

```
  \xdef\@law@temp{\the\@ltok@citelast}%
  \xdef\@law@temptwo{\the\@ltok@usercitelast}%
  \ifx\@law@temp\@law@temptwo%
  \else%
    \the\@ltok@ptoctwo%
    {\@law@barinfull\the\@ltok@citelast}%
  \fi%
    \@law@shiftparas%
  \endgroup}%
```

`\@law@shortrecurse` This macro executes until all of the citation details have been printed, including all parallels.

```
% [RECURSING MACRO GOES HERE!]
```

`\@law@shortprint`

```
\def\@law@shortprint{%
% If {\sc Camel} sees more than one pinpoint page or section,
% the bridges preceding
% the page references must be set to their plural form.
% This change applies to both long and short form citations.
% It has to be cloned in each, however, because the decision has
% to be taken with respect to each cite in a string of parallels.
%    \begin{macrocode}
  \begingroup%
  \@law@fetchparas%
  \@law@tidybridges%
```

The actual recursive printing works.

```
      \global\ltokspecialface=\@ltok@citefirstoptionface%
      {\the\@ltok@citefirstmainface%
      \@law@barinshort\the\@ltok@citefirst}%
```

If the pinpoint reference is empty, we tack on the location page after the appropriate bridge. Otherwise, we do nothing for the present; any pinpoint reference will be produced later on. Note that the bridge used here will only be in plural form if something was given for use as a pinpoint reference.

```
      \the\@ltok@conetop%
      \ifcat$\the\@ltok@argtwo$%
        \the\@ltok@citepage%
      \else%
        {\@law@barkill\the\@ltok@argtwo}%
      \fi%
    \@law@shiftparas%
  \endgroup}%
```

Print format subroutines

`\@law@casepreformat` This is empty; no special preparations are necessary for the printing of a case citation.

```
\gdef\@law@casepreformat{}%
```

`\@law@statutepreformat` If short statutory references are in force, the title and the following bridge are set to nil, if some reference is provided in the first citation part.

```
\gdef\@law@statutepreformat{%
\if@law@statuteverbose%
\else%
  \ifcat$\the\@ltok@citefirst$%
  \else%
   \global\@ltok@name{}%
   \global\@ltok@ttocone{}%
  \fi%
\fi%
}%
```

`\@law@articlepreformat` If a work by the identical author has been cited more than once, we leave everything intact (the print routine does the necessary culling); otherwise, we cut the title here. The use of `\@law@authortracing` is explained above.

```
\gdef\@law@articlepreformat{%
\ifcat$\the\@ltok@author$%
\else%
{\@law@clean\@ltok@author\@law@authortracing%
\expandafter\expandafter\expandafter\if\expandafter%
\csname\@law@authortracing\endcsname2%
 \else%
  \global\@ltok@atot{}\global\@ltok@name{}%
\fi}\fi}%
```

\@law@bookpreformat This provides the same treatment given to articles.

```
\gdef\@law@bookpreformat{%
\ifcat$\the\@ltok@author$%
\else%
{\@law@clean\@ltok@author\@law@authortracing%
\expandafter\expandafter\expandafter\if\expandafter%
\csname\@law@authortracing\endcsname2%
 \else%
  \global\@ltok@atot{}\global\@ltok@name{}%
 \fi}\fi}%
```

⟨/lawcitestyle⟩

2.3 Camel citation formats

The style definitions are stored in a separate file, to make it easier and less risky for users to play with the styles to produce desired output. Table 3 provides a guide to the macro arguments and their functions.

2.3.1 Books

The styles used for citing books and book-like things follow.

⟨∗lawcite⟩

```
\ProvidesFile{law.cit}[1994/12/07]
\newcitestyle{book}%
 {srsrrrB}
 {[a],\ [t][c]\ [p](pl)\ [rp]\ [e]:[id]\ at~(pl)\ at~[xrf]}
%
\newcitestyle{booklet}%
 {riririB}
 {[a],\ [t][c]\ [p](pl)\ [rp]\ [e]:[id]\ at~(pl)\ at~[xrf]}
%
\newcitestyle{techreport}%
 {riririB}
 {[a],\ [t][c]\ [p](pl)\ [rp]\ [e]:[id]\ at~(pl)\ at~[xrf]}
%
\newcitestyle{mastersthesis}%
 {riririB}
 {[a],\ [t][c]\ [p](pl)\ [rp]\ [e]:[id]\ at~(pl)\ at~[xrf]}
```

2.3.2 Articles

Styles used for articles and similar shortish things follow. They're all the same, but the clones help keep things clear when following the action between LaTeX and BibTeX.

```
\newcitestyle{article}%
 {rsirsrA}
 {[a],\ [t],\ [c]\ [p],\ (pl),\ [rp]\ [e]:[id]\ at~(pl)\
at~[xrf]}
```

```
\newcitestyle{incollection}%
 {rsirsrA}
 {[a],\ [t],\ [c]\ [p],\ (pl),\ [rp]\ [e]:[id]\ at~(pl)\
at~[xrf]}
\newcitestyle{inbook}%
 {srirsrA}
 {[a],\ [t],\ [c]\ [p],\ (pl),\ [rp]\ [e]:[id]\ at~(pl)\
at~[xrf]}
```

2.3.3 Cases

There are several styles for cases; we pretty much need a separate style for each jurisdiction.

```
\newcitestyle{case}%
 {rrirrsC}
 {[a][t],\ [c]\ [p],\ (pl),\ [rp]\ [e]:[id]\ at~(pl)\ at~[xrf]}
```

2.3.4 Statutes

There are several of these. More may need to be added on an *ad hoc* basis.

```
\newcitestyle{statute}%
 {rrrsrsS}
 {[a][t],\ [c]\ \S~[p](pl)\ \S\S~[rp]\ [e]:[id]\ \S~(pl)\
\S\S~[xrf]}
%
\newcitestyle{jstatute}%
 {rrrsrsS}
 {[a][t][c]\ \S~[p](pl)\ \S\S~[rp]\ [e]:[id]\ \S~(pl)\
\S\S~[xrf]}
⟨/lawcite⟩
```

3 Extraction utilities

3.1 The Driver

Here is a simple driver for extracting the files in the package.

```
⟨*installer⟩
\def\batchfile{law.ins}
\input docstrip.tex
\preamble

Copyright (C) 1992--95 Frank Bennett, Jr.
All rights reserved.

This file is part of the Law module for the Camel package.
\endpreamble

\def\batchfile{camel.dst}      % ignored in distribution
```

```
\input docstrip.tex              % ignored in distribution

\keepsilent

\preamble

This file is part of the Law module of the Camel package.
-----------------------------------------------------------
This is a generated file.

IMPORTANT NOTICE:

You are allowed to change this file, subject to the following
conditions.  Under any circumstances, new macro definitions
should not be added to this file.  You are welcome to modify
the macro definitions contained in this file for your own
use.  If you pass a copy of the modified version to someone
else, you should (a) let me know about the change on
fb@soas.ac.uk, and (b) put a note of the changes and of your
own contact details in the file.  Furthermore, you must
acknowledge Camel and its author(s) in the new file (if it
is distributed to others), and you must attach these same
conditions to the new file.

You are not allowed to distribute this file alone.  You are not
allowed to take money for the distribution or use of this file
(or a changed version) except for a nominal charge for copying
etc.

You are allowed to distribute this file under the condition that
it is distributed with all of its contents, intact.

For error reports, or offers to help make Camel a more powerful,
friendlier, and better package, please contact me on
'fb' at soas.ac.uk

\endpreamble


\Msg{*** Generating Camel style file (.cst) ***}

\generateFile{law.cst}    {t}{\from{law.dtx}{lawcitestyle}}


\Msg{*** Generating Camel citation format file (.cst) ***}

\generateFile{law.cit}    {t}{\from{law.dtx}{lawcite}}


\Msg{*** Generating Camel BibTeX style file (.bst) ***}
```

```
\postamble
\endpostamble

\generateFile{law.bst}    {t}{\from{camel.dtx}{bstheader}
                              \from{law.dtx}{bstheader}
                              \from{camel.dtx}{bstlibrary}
                              \from{law.dtx}{bstfunctions}
                              \from{camel.dtx}{bsttrailer}}

\keepsilent


\ifToplevel{
\Msg{************************************************************}
\Msg{*}
\Msg{* To finish the installation you have to move the following}
\Msg{* file into a directory searched by TeX:}
\Msg{*}
\Msg{* \space\space law.cst}
\Msg{* \space\space law.cit}
\Msg{*}
\Msg{* You should also move the following file into a directory}
\Msg{* searched for style files by BibTeX:}
\Msg{*}
\Msg{* \space\space law.bst}
\Msg{*}
\Msg{* Other style modules can be found on CTAN in the 'styles'}
\Msg{* subdirectory below Camel itself.}
\Msg{*}
\Msg{************************************************************}
}
```
⟨/installer⟩

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.

Change History

LexiBib1.0c

`"General"`: Added support for Japanese statutes. 14

0.1a

`"General"`: Function `type.last.char` and `\bridges` declaration used to handle titles ending in a numeral correctly (by the insertion of a comma). Added to the `@article` and @book entries as a trial; will propagate to other entry types once this change is trusted. 5

1.0g

`"General"`: Added the 'techreport' function, to support draft article sent to Pedro Aphalo for comments. 11

2.0c

`"General"`: Moved `\@law@forcingfalse` outside of a conditional expression at the end of the print routine, to correct the failure of the forcing mechanism to work more than once. 19

2.0l

`"General"`: Fairly drastic simplification of the forcing mechanism, in the course of providing for unified handling of citation strings. Code now much easier to follow. 19