Table of Authorities (`toa.dtx`)

Charles Duan

March 17, 2025

Legal briefs and other documents often include a Table of Authorities, listing all of the references cited and the pages where those citations occur. One of the motivations for creating this package was the automatic generation of complete, correct, and properly formatted Tables of Authorities.

The table is generated using LaTeX's auxiliary file feature. Citations generate index lines that are written to a `\jobname.toa` file, and those lines are read, sorted, and formatted to produce the table. Indexing can be disabled with `\disabletoa` and enabled with `\enabletoa`.

A Table of Authorities is a complex structure, because it is typically sorted into sections based on reference types. This section describes how references are sorted into lists, how those lists get their headings, how page numbers in the tables are displayed, and finally how the table is formatted.

```
\newif\if@hi@in@toa
\newif\if@hi@toa@disabled
\let\disabletoa\@hi@toa@disabledtrue
\let\enabletoa\@hi@toa@disabledfalse
```

1 Table of Authorities Categories

Each reference type is associated with a category, used to organize the Table of Authorities. The category can be altered using the reference definition parameter toacat or by the `\SetTOACategory` macro. The categories are used to assemble the headings for the table.

Specifically, every category has three properties:

1. A name for internal use.
2. A "supercategory" that associates the category with others that this category should be joined with in the Table of Authorities. For example, the categories for statutes and regulations should have the same supercategory, so that they are placed in a single section with the Table of Authorities. Supercategories are presented in alphabetical order.
3. A textual description of the category, for use in constructing the Table of Authorities heading. Because both the singular and plural forms are

needed, the textual description consists of two to three parts, separated by slashes:

(a) The base text common to both forms.

(b) [*Optional*] Text to append to the singular form only.

(c) Text to append to the plural form only.

A new TOA category may be defined with `\NewTOACategory`. The macro takes four parameters, the first three corresponding to the three properties of categories above, and the last (being "Y" or "N") indicating whether to include the supercategory in the default table of authorities.

TODO: The underlying implementation here could be improved to rely less on marker symbols. Also, categories should have a precedence number indicating the order in which they should appear in the TOA heading.

```
\def\NewTOACategory#1#2#3#4{%
    \@namedef{hi@cats@#1}{#2:#3}%
    \ifx#4Y%
        \AddToList{hi@toacat@supercats}{#2}%
    \fi
}
```

1.1 Setting and Getting Categories

Each reference type may have a default category, set with the `\DefaultTOACategory` macro. The parameters are #1 the reference type, and #2 the category name. If the category name is blank, then there will be no default category. This default category will be used unless another category is specified for a particular reference, overriding the default category.

```
\def\DefaultTOACategory#1#2{%
    \ifstrempty{#2}{%
        \cslet{hi@catd@#1}\relax
    }{%
        \@ifundefined{hi@cats@#2}{%
            \PackageError\hi@pkgname{%
                TOA category #2 does not exist, in setting the\MessageBreak
                default for reference type #1%
            }{Define TOA category #2 first with \string\NewTOACategory}%
        }{%
            \@namedef{hi@catd@#1}{#2}%
        }%
    }%
}
```

The `\SetTOACategory{`⟨*reference*⟩`}{`⟨*category*⟩`}` macro sets the TOA category for ⟨*reference*⟩. This macro should be called within the reference definer macros to set the category, or it is invoked by the toacat parameter. If it is called multiple times, then the last category will prevail. If it is not called, then the reference will not appear in the table. If ⟨*category*⟩ is empty, then any previously set category will be removed.

```
\def\SetTOACategory#1#2{%
    \ifstrempty{#2}{%
        \global\cslet{hi@cat@#1}\relax
    }{%
        \@ifundefined{hi@cats@#2}{%
            \PackageError\hi@pkgname{%
                Invalid Table of Authorities category #1,\MessageBreak
                given for reference #2%
            }{%
                Please enter a valid Table of Authorities category.
            }%
        }{%
            \global\csletcs{hi@cat@#1}{hi@cats@#2}%
        }%
    }
}
```

Retrieves the TOA category for a reference `#1`, and executes a callback `#2` with the category text as an argument. If the reference has no category, then the callback will not be executed.

```
\def\hi@toacat@get#1#2{%
    \@ifundefined{hi@cat@#1}{}{%
        \@expand{#2}{\csname hi@cat@#1\endcsname}{ii}%
    }%
}
```

Transfers a reference's category to another reference. #1 is the reference with the already-defined category; #2 is the reference to receive the category. The category for #1 is deleted.

```
\def\hi@toacat@transfer#1#2{%
    \global\csletcs{hi@cat@#2}{hi@cat@#1}%
    \global\cslet{hi@cat@#1}\relax
}
```

Sets a reference's TOA category to the reference type's default category. #1 is the reference name, and #2 the reference type. The category will not be changed if it is already set, or if the reference type has no default category.

```
\def\hi@toacat@setdefault#1#2{%
    \@ifundefined{hi@cat@#1}{%
        \@ifundefined{hi@catd@#2}{}{%
            \SetTOACategory{#1}{\csname hi@catd@#2\endcsname}%
        }%
    }{}%
}
```

1.2 Supercategories

Supercategories are stored in a sorted list (see `sortlist.dtx`), which is sorted alphabetically.

```
\NewSortedList{hi@toacat@supercats}{\SortEasyAlpha}{\StripForAlpha{#1}{\def#2}}
\ListElementsMustBeUnique{hi@toacat@supercats}
```

1.3 Predefined Categories

Definitions of the categories follow in the source code, but they are used to produce a table in the documentation, so please consult the package file for the code.

The following are the TOA categories, with their supercategories and display texts as described above. Note the choice of supercategory names, which coincidentally sort alphabetically in the correct order. Because the actual supercategory names are never displayed, any new supercategories can be named to fit into the existing ordering.

Category	Supercat.	Text	In TOA?
other	other	Other Source/s	Y
const	const	Constitutional Provision/s	Y
found	const	Foundational Document/s	Y
statute	edict	Statute/s	Y
regulation	edict	Regulation/s	Y
rule	edict	Rule/s	Y
case	case	Case/s	Y
admin	case	Administrative Decision/s	Y
treaty	edict	Treat/y/ies	Y
abbrev	abbrev	Abbreviation/s	N

This macro will perform a function for each of the Table of Authorities groups, in the proper order. The function should be given as a macro definition in #1, the macro definition accepting one argument which will be the supercategory name.

```
\long\def\hi@toacat@each#1{%
    %
    % Unfortunately, this macro is used to iterate over items in the TOA, which
    % also stores items in SortedLists. So we can't just use \ShowList directly
    % since \ShowList can't be used in a nested fashion. So instead, we iterate
    % over the supercategory list to construct a series of macro expansions of
    % the given parameter, and then execute that series.
    %
    \begingroup % Will be ended when \reserved@a runs
    \def\hi@toacat@temp{\endgroup}%
    \def\hi@toacat@do##1{#1}%
    \ShowList{hi@toacat@supercats}{%
        \@expand{\appto\hi@toacat@temp}{\hi@toacat@do{##1}}{i}%
    }%
    \hi@toacat@temp
}
```

2 Altering Categories in a Document

It may be useful to alter the TOA categories in a document, to create new categories or recategorize certain references or reference types. The macros described above allow for that. The following is a how-to guide on achieving a few such effects.

Place one reference in a different part of the TOA: use the toacat parameter when defining that reference.

Remove a reference from the TOA: use the notoa parameter, or set the toacat parameter to blank.

Place all references of a given type in a different part of the TOA (e.g., treat modelcode references like statutes): Use

\DefaultTOACategory{⟨*type*⟩}{⟨*category*⟩}

Example:
```
\DefaultTOACategory{modelcode}{statute}
```

where ⟨*category*⟩ is chosen from the table above.

Add a new TOA section (e.g., put film references in their own section): Use

```
\NewTOACategory{⟨category⟩}{⟨super⟩}{⟨text⟩}{Y}
\DefaultTOACategory{⟨type⟩}{⟨category⟩}
```

Example:

```
\NewTOACategory{m}{movies}{Motion Picture/s}{Y}
\DefaultTOACategory{film}{m}
```

where ⟨*category*⟩ and ⟨*super*⟩ are new names (not in the above table), and ⟨*text*⟩ is the text of the heading.

Add a new type to an existing TOA section (e.g., add “Model Code” to Statutes and Regulations): Use

```
\NewTOACategory{⟨category⟩}{⟨super⟩}{⟨text⟩}{Y}
\DefaultTOACategory{⟨type⟩}{⟨category⟩}
```

Example:

```
\NewTOACategory{mc}{edict}{Model Code/s}{Y}
\DefaultTOACategory{modelcode}{mc}
```

where ⟨*category*⟩ is a new name (not in the above table), {⟨*super*⟩} is the supercategory name corresponding to the existing TOA section, and ⟨*text*⟩ is the text of the heading.

3 Indexing References

References are indexed for the Table of Authorities upon every citation, by having entries added to the `\jobname.toa` file automatically for each citation command. The command `\addtotoa` described in `iface.dtx` can also add an entry manually.

An interesting challenge for indexing is that, especially for briefs, references may be cited in parts of a document that use different pagination systems. As a result, it is necessary to store detailed information about where a reference is cited in a document in order to produce a correct listing of pages in the table.

Five pieces of information are written out to the file for each use of a reference:

1. The reference name, and possibly the page for references where the page number is included in table of authorities entries.

2. The page number as formatted.

3. The page number as an arabic number.

4. The prevailing definition of `\thepage`, so that different page counting schemes can be identified.

5. A comma-separated list of “tags” under which the index entry should be categorized. Typically this will be up to two tags, one based on the reference’s category and one based on the reference’s location. A tag is a colon-separated structure comprising a supercategory name and a textual heading name (corresponding to the TOA Categories elements described in `toa.dtx`).

Proposes adding a reference for indexing in the table of authorities. #1 is the reference, and #2 is the page number if any. Even if a reference is proposed for adding, the following conditions must be true:

- The TOA is not disabled
- The citation containing the reference to be added to the TOA is not itself in the TOA
- The reference has an associated category (see `toa.dtx`).

```
\def\hi@addtotoa#1#2{%
    \hi@toacat@get{#1}{\hi@addtotoa@{#1}{#2}}%
}
%
% Called only if the reference had a category. |#1| is the reference name, |#2|
% the page number, |#3| the category text.
\def\hi@addtotoa@#1#2#3{%
    \if@hi@in@toa\else \if@hi@toa@disabled\else
        \addtocontents{toa}{%
            \protect\contentsline{hi@toa}{%
                #1\@ifundefined{tcpg@#1}{}{\ifstrempty{#2}{}{ at #2}}%
            }{%
                {%
                    \thepage
                }{%
                    \noexpand\the\noexpand\c@page
                }{%
                    % We want this to run at the time the |\write| is
                    % executed.
                    \noexpand\expandonce\thepage
                }{%
                    #3%
                    \ifx\hi@toa@locationtag\relax\else
                        ,\hi@toa@locationtag
                    \fi
                }%
            }{}%
        }%
    \fi\fi
}
```

Declares that the page number is relevant and should be included for a citation. This command should be issued by reference macro definers where the reference type ought to include a page number in the TOA.

```
\def\hi@include@page@in@toa#1{%
    \global\@namedef{tcpg@#1}{}%
}
```

4 Reading the Index

Prior to generating a table of authorities, the index file is read and its entries are sorted to produce three sets of data structures:

- For each supercategory, a list of references falling within that supercategory.
- For each supercategory, a list of textual headings for that supercategory.
- For each reference, a macro identifying the page numbers associated with that reference.

Sets up the Table of Authorities. This reads the `toa` file, performs all the reference use counting, and opens the `toa` file for writing. It then undefines itself so it cannot be called again. It should be run before displaying any TOA entries.

```
\def\hi@setuptoa{%
    \@hi@in@toatrue
    \global\let\@last@toatitle\relax
    \hi@toa@counter\z@
    \@input{\jobname.toa}%
    \@hi@in@toafalse
    \if@filesw
        \newwrite\tf@toa
        \immediate\openout\tf@toa \jobname.toa\relax
    \fi
    \global\let\hi@setuptoa\relax
}
```

This is run for each entry in the `toa` file, as produced above. The arguments (after expansion of the groups in the second argument) match the five elements given above with respect to `\hi@addtotoa@`.

The macro invokes `\hi@toa@addpg` to update the page number listing for the reference, and invokes `\hi@toa@addtolist` to add the reference's information to the supercategory lists.

```
\def\l@hi@toa#1#2{\l@hi@toa@{#1}#2}
\def\l@hi@toa@#1#2#3#4#5{%
    \expandafter\hi@toa@addpg
        \csname hi@pg@\detokenize{#1}\endcsname
        {#2}{#3}{#4}%
    \@for\hi@toa@tmp:=\@empty#5\do{%
        \@expand{\find@in{:}}\hi@toa@tmp{i}{\hi@toa@addtolist{#1}}{%
            \hi@toa@addtolist{#1}\hi@toa@tmp\hi@toa@tmp
        }%
    }%
}
\make@find@in{:}
```

A counter for TOA items, just for informational purposes.

```
\newcount\hi@toa@counter
```

4(a) Adding to Lists

Adds a given reference to a supercategory list, called `hi@toa@`⟨*supercat*⟩. If the list does not already exist, then create it. The list is alphabetically sorted and its items are unique.

Also updates the heading lists for the supercategory by invoking `\hi@toa@headlist@add`.

The message counters are updated and displayed here because we want to track unique additions of references to TOA lists.

#1 is the reference, #2 the supercategory, #3 the textual description.

```
\def\hi@toa@addtolist#1#2#3{%
    \IfKnownList{hi@toa@#2}{}{%
        \NewSortedList{hi@toa@#2}\SortEasyAlpha{%
            \hi@toa@makesortable{##1}{\def##2}%
        }%
        \ListElementsMustBeUnique{hi@toa@#1}%
    }%
    \IfListContains{hi@toa@#2}{#1}{}{%
        \AddToList{hi@toa@#2}{#1}%
        \hi@toa@headlist@add{#2}{#3}%
        \advance\hi@toa@counter\@ne
        \message{(\the\hi@toa@counter)}%
    }%
}
```

Given #1 a reference name and #2 a callback, turns the TOA citation form of #1 into sortable text and runs #2 on it.

```
\def\hi@toa@makesortable#1#2{%
    \find@in{ at }{#1}{\hi@toa@makesortable@}{\hi@toa@makesortable@{#1}{}}{#2}%
}
```

#1 is the reference name, #2 the page number, #3 the callback.

```
\def\hi@toa@makesortable@#1#2#3{%
    \@expand\hi@toa@makesortable@@{%
        \csname\@ifundefined{lc@#1}{fc@#1}{lc@#1}\endcsname
    }{ii}{#2}{#3}%
}
```

#1 is the defined text of the reference macro, #2 the page number, #3 the callback.

```
\def\hi@toa@makesortable@@#1#2#3{%
    \uppercase{\StripForAlpha{#1#2}}{#3}%
}
\make@find@in{ at }
```

4.1 Heading Lists

For each supercategory, the textual description heading may comprise one or more types of references. For example, the "edict" supercategory may have statutes and/or regulations, and the heading should reflect the particular items to be listed. So if one regulation and several statutes have been cited, then the heading should read "Statutes and Regulation," for example.

To implement this, for every supercategory there is a "heading list" associated with the supercategory. The heading list contains every textual description associated with the supercategory, plus a flag for each textual description for whether it should be singular or plural. The first time a textual description is encountered for a heading list, the singular form is used. If the same description is encountered again, then it is updated to be marked as plural.

`\hi@headlist@`⟨*supercat*⟩ is the macro for the heading list. That macro contains a list of paired items, one for each textual description encountered:

- A macro `\hi@toa@headlist@singular` or `\hi@toa@headlist@plural`
- The given textual description

Determines if the heading list exists for the given supercategory. If not, creates it. Otherwise, searches for the textual description in the list and updates it. #1 is the supercategory name, #2 is the textual description.

```
\def\hi@toa@headlist@add#1#2{%
    \expandafter\hi@toa@headlist@add@\csname hi@headlist@#1\endcsname{#2}%
}
```

#1 is the `\hi@headlist@`⟨*supercat*⟩ macro, #2 is the textual description. `\reserved@a` will be the textual description to search for.

```
\def\hi@toa@headlist@add@#1#2{%
    \ifx#1\relax
        \def#1{\hi@toa@headlist@singular{#2}}%
    \else
        \def\reserved@a{#2}
        \expandafter\hi@toa@headlist@update#1\@stop{}#1%
    \fi
}
```

Iterates through the heading list, searching for textual description saved in `\reserved@a`. #1 is the singular/plural marker, #2 the textual description in the list being considered, #3 is the remaining list, #4 the part of the list already searched, #5 the heading list macro.

```
\def\hi@toa@headlist@update#1#2#3\@stop#4#5{%
    \def\reserved@b{#2}%
    \@test \ifx\reserved@a\reserved@b\fi {%
        \def#5{#4\hi@toa@headlist@plural{#2}#3}%
    }{%
        \ifblank{#3}{%
            \edef#5{%
                \unexpanded{#4#1{#2}\hi@toa@headlist@singular}%
                {\expandonce\reserved@a}%
            }%
        }{%
            \hi@toa@headlist@update#3\@stop{#4#1{#2}}{#5}%
        }%
    }
}
```

As noted above in `toa.dtx`, textual descriptions use a convention of slashes to differentiate between the singular and plural forms. The `\hi@toa@headlist@singular` and `\hi@toa@headlist@plural` macros parse the slashes to construct the desired form, and also insert the relevant punctuation for the subsequent element in the heading list.

```
\def\hi@toa@headlist@singular#1{%
    \find@in{/}{#1}{\hi@toa@headlist@choose\@firstoftwo}{#1}%
    \hi@toa@headlist@lookahead
}
\def\hi@toa@headlist@plural#1{%
    \find@in{/}{#1}{\hi@toa@headlist@choose\@secondoftwo}{#1}%
    \hi@toa@headlist@lookahead
}
\make@find@in{/}
\def\hi@toa@headlist@lookahead#1{%
    \ifx#1\@empty\else
        \expandafter\hi@toa@headlist@lookahead@\expandafter#1%
    \fi
}
\def\hi@toa@headlist@lookahead@#1#2#3{%
    \@test \ifx#3\@empty \fi{%
        \space and\space #1{#2}#3%
    }{%
        ,\space#1{#2}#3%
    }%
}
\def\hi@toa@headlist@choose#1#2#3{%
    #2\find@in{/}{#3}{#1}{#1{}{#3}}%
}
```

Shows a heading list, for example in a heading.

```
\DeclareRobustCommand\hi@toa@headlist@show[1]{%
    \csname hi@headlist@#1\endcsname\@empty
}
```

4.1(a) Location-Based Lists It may be desirable to prepare a chapter-by-chapter bibliography, such that references are tagged not by type but by location in the document. The macro `\AuthorityTag{`⟨*tag*⟩`}` will tag all subsequent references cited with the given ⟨*tag*⟩, which can then be used as a list of the Table of Authorities.

The current location tag is stored in `\hi@toa@locationtag`.

```
\def\AuthorityTag#1{%
    \def\hi@toa@locationtag{#1}%
}
\let\hi@toa@locationtag\relax
```

5 Composing Page Number Lists

If a reference is cited on pages 3, 4, 5, 7, 8, and 10, it is desirable to group these numbers into compact range forms for presentation in the Table of Authorities: "3–5, 7–8, 10." The package does so by reading the page numbers in order, keeping a running compilation of the properly formatted aggregate number listing and revising the last element of that listing as necessary.

The `\hi@pg@`⟨*ref*⟩ macros take on the following form:

{⟨*last-display-page*⟩} {⟨*last-numeric-page*⟩} {⟨*last-*`\thepage`⟩} {⟨*built-page-list*⟩}

where appending ⟨*last-display-page*⟩ to ⟨*built-page-list*⟩ will give the complete page list so far. Also, ⟨*built-page-list*⟩ is prepended with a list of `*`s to indicate the number of citations to a particular reference, for counting purposes.

#1 is the `\hi@pg@`⟨*ref*⟩ macro. #2-#4 are the elements from the `toa` file listing: the display-form page number, the numeric page number, and the definition of `\thepage`.

```
\def\hi@toa@addpg#1#2#3#4{%
    \@test \ifx#1\relax\fi {%
        \def#1{{}{#3}{#4}{*:#2}}%
    }{%
        \expandafter\hi@toa@addpg@#1#1{#2}{#3}{#4}%
    }%
}
```

Updates the running list of pages, when given a new page number.

#1-#4 are the four hi@pg@⟨*ref*⟩ parameters, which are described above, #5 is `\hi@pg@`⟨*ref*⟩, #6-#8 are #2-#4 of `\l@hi@toa`.

```
\def\hi@toa@addpg@#1#2#3#4#5#6#7#8{%
    \def\reserved@a{#3}\def\reserved@b{#8}%
    \ifx\reserved@a\reserved@b
```

For the same `\thepage`: Compare the last-added page number to the page number to be added. If they are the same, then just increment the count. If the new page number is one greater, then see if we're already working with a range (#1, ⟨*last-display-page*⟩ is not blank). If so, then just increment the count. Otherwise, increment the count and add a dash. In any event, set ⟨*last-display-page*⟩ to the new display page number.

```
        \@tempcnta=#2\relax
        \ifnum#7=\@tempcnta
            \def#5{{#1}{#2}{#3}{*#4}}%
        \else
            \advance\@tempcnta\@ne
            \ifnum#7=\@tempcnta
                \ifblank{#1}{%
                    \def#5{{#6}{#7}{#8}{*#4--}}%
                }{%
                    \def#5{{#6}{#7}{#8}{*#4}}%
                }%
            \else
```

If the numeric increase is greater than one, then compose the full aggregate list (#4#1), and tack on a comma and the new display number.

```
                \def#5{{}{#7}{#8}{*#4#1, #6}}%
            \fi
        \fi
    \else
```

If `\thepage` has changed, then compose the full list and tack on a comma and the new number.

```
        \def#5{{}{#7}{#8}{*#4#1, #6}}%
    \fi
}
```

5(a) Using Page Number Lists

Composes the page number list and runs the callback on it. #1 is the reference name, #2 is a callback.

```
\def\hi@toa@usepg#1#2{%
    \@expand{\@unbrace\hi@toa@usepg@}{%
        \csname hi@pg@\detokenize{#1}\endcsname
    }{ii}{#2}%
}
% \#1-\#4 are the elements of the |\hi@pg@|\meta{ref} structure, \#5 is the
% callback.
\def\hi@toa@usepg@#1#2#3#4#5{%
    \find@in{:}{#4#1}{%
        \@tworun\@gobble\hi@toa@usepg@passim
    }{%
        \PackageError\hi@pkgname{%
            Page data structure `#4#1' contains \MessageBreak
            no colon, but it should. This indicates \MessageBreak
            a bug in the hicite program%
        }{Check the hicite program.}%
        \@gobble
    }{#5}%
}
\make@find@in{:}
% \#1 is the page list, \#2 is the callback.
\def\hi@toa@usepg@passim#1#2{%
    \@test \ifnum\c@passimnum>\z@ \fi{%
```

```
        \hi@toa@usepg@passim@{1}#1, \@stop{#1}{#2}%
    }{#2{#1}}%
}
\def\hi@toa@usepg@passim@#1#2, #3\@stop#4#5{%
    \@test \ifnum#1<\c@passimnum \fi{%
        \ifstrempty{#3}{#5{#4}}{%
            \@expandarg\hi@toa@usepg@passim@{%
                \number\numexpr#1+1\relax
            }#3\@stop{#4}{#5}%
        }%
    }{#5{\PassimText}}%
}
```

Counts the number of times a reference was used, based on the `\hi@pg@`⟨*ref*⟩ data. #1 is ⟨*ref*⟩, #2 a callback. It is entirely expandable.

```
\def\hi@toa@refcount#1#2{%
    \@expand{\@unbrace\hi@toa@refcount@}{\csname hi@pg@#1\endcsname}{ii}{#2}%
}
\def\hi@toa@refcount@#1#2#3#4#5{%
    \find@in{:}{#4}{\hi@toa@refcount@@}{%
        \PackageError\hi@pkgname{%
            Reference count data `#4' should have contained a \MessageBreak
            colon, but it does not. This probably indicates \MessageBreak
            a programming error.
        }{Check the hicite program.}%
        \@gobble
    }{#5}%
}
\def\hi@toa@refcount@@#1#2#3{\hi@toa@refcount@@@{1}#1\@stop{#3}}%
\def\hi@toa@refcount@@@#1#2#3\@stop#4{%
    \ifstrempty{#3}{#4{#1}}{%
        \@expandarg\hi@toa@refcount@@@{\number\numexpr#1+1\relax}#3\@stop{#4}%
    }%
}
```

5(b) Starred References

Some courts ask litigants to insert asterisks in the Table of Authorities before references that they rely most heavily on. This can be done automatically by counting the number of times each reference has been used.

The macro `\StarCount{`⟨*num*⟩`}` sets the minimum number of uses of a reference before that reference receives a star. The package option toastar sets the value to 4. By default it is zero, in which case no stars are displayed.

The macro `\StarText` can be defined as text to place at the beginning of the Table of Authorities explaining the stars.

The macro `\StarMark` is the actual text of the star; it will be added before the relevant TOA entries via `\everypar`.

Adds a star before the reference if its usage count exceeds the minimum count. #1 is the reference name. The star is added by an `\everypar` because TOA entries are initially in vertical mode.

```
\newcount\hi@toa@starcount
\def\hi@toa@star#1{%
    \ifnum\hi@toa@starcount>\z@
        \ifnum\hi@toa@refcount{#1}\@iden<\hi@toa@starcount\else
            \everypar{\StarMark\everypar{}}%
        \fi
    \fi
}
\def\StarCount#1{\hi@toa@starcount=#1\relax}
\def\StarText{%
    Authorities upon which this brief primarily relies are indicated with
    asterisks in the margin.\par
}
\def\StarMark{\llap{* }}
```

6 Showing the Table

The Table of Authorities is produced by calling `\tableofauthorities`. This will create one section for each reference category as described in `refs.dtx`. Each heading will be created using whatever `\toaheading` is defined as.

```
\def\tableofauthorities{%
    \ifnum\hi@toa@starcount>\z@
        \StarText
    \fi
    \begingroup
        \hi@toacat@each{%
            \AuthorityTable[%
                \toaheading{\hi@toa@headlist@show{##1}}\par
                \ifdim\lastskip<\toaskip
                    \vskip-\lastskip
                \else
                    \vskip-\toaskip
                \fi
                \parindent\z@
                \parskip=\toaskip
            ]{##1}%
        }%
    \endgroup
}
\def\toaheading{\section*}
```

Alternately, the macro `\AuthorityTable[`⟨*heading-text*⟩`]{`⟨*list-name*⟩`}` will produce a single Table of Authorities list. This can be used for more fine-grained control over what tables are displayed.

#1 is optional text to be displayed before the table if the table has any content. #2 is the table name.

```
\newcommand\AuthorityTable[2][]{%
    \hi@setuptoa
    \@hi@in@toatrue
    \noid
    \IfKnownList{hi@toa@#2}{\IfEmptyList{hi@toa@#2}{}{%
        #1%
        \ShowList{hi@toa@#2}{%
            \global\let\@this@toatitle\relax
            \hi@toa@usepg{##1}{\@toaline{\hi@toa@star{##1}\toacite{##1}}}%
            \global\let\@last@toatitle\@this@toatitle
        }%
    }}{}%
    \@hi@in@toafalse
}
```

Displays one reference's line in the Table of Authorities. #1 is the text of the reference citation, #2 the page number aggregate.

```
\def\@toaline#1#2{%
    {\hi@toa@par@settings
    #1\nobreak
    \TOALeader
    \nobreak
    {\normalfont \normalcolor \hi@toa@brkpg{#2}}%
    \par}%
}
```

Sets up the paragraph margin settings for TOA entries.

```
\def\hi@toa@par@settings{%
    \leftskip\z@\rightskip \@tocrmarg
    % Make the right margin ragged. This is necessary because the ditto spacing
    % gets messed up on occasion due to space stretching.
    \rightskip=1\rightskip plus .1\hsize
    \parfillskip -\rightskip
    \hangindent\toahangindentlen\relax
    \parindent \z@\@afterindenttrue
    \interlinepenalty\@M
}
```

Inserts the dots for TOA entries.

```
\def\TOALeader{%
    \leaders\hbox{%
```

```
        $\m@th \mkern \@dotsep mu\hbox{.}\mkern \@dotsep mu$%
    }\hskip 2em \@plus 1fill \@minus 0.5em
}
```

Displays a single entry in the Table of Authorities. This is essentially another citation command like `\sentence` except it invokes less overhead because there is no punctuation at the end of the entry to check.

```
\def\toacite#1{%
    \begingroup
    \let\clause\hi@clause
    \let\sentence\hi@sentence
    \hi@pse@parse{#1}{\hi@draw@citation{\@hi@captrue\@hi@senttrue}{}}%
    \endgroup
}
```

On occasion, the best presentation of the TOA entry involves breaking the page number listing across a line. This macro computes the best place for doing so.

The basic idea is that, for each page number item, three glue entries are inserted: $-$`\rightskip`, G_a and G_b. G_b has natural dimension `\hsize` but is shrinkable to zero, and $G_a = -G_b +$ `\rightskip`. G_a is inserted with a penalty that favors breaking after the last possible comma, but G_b cannot be broken beforehand.

If no break occurs between two page numbers, then the total glue is $-$`\rightskip` $+ G_a + G_b = 0$. If a break is permitted, though, then G_a is discarded. As a result, the preceding page number gets glue $-$`\rightskip`, pulling it all the way to the right margin, and the next page number starts a new line and gets glue G_b, pushing it as far to the right margin as possible and (because of the shrinkage) causing the line-breaking algorithm to favor putting as few items on this new line as possible.

Below, `\@tempskipa` is G_a, `\@tempskipb` is G_b, and `\@tempcnta` is a descending penalty placed before G_a.

```
\def\hi@toa@brkpg#1{%
    \@tempcnta=9999\relax
    \@tempskipb=\hsize\@minus\hsize\relax
    \@tempskipa=\glueexpr \@tempskipb * -1 + 1\rightskip\relax
    \find@in{, }{#1}\hi@toa@brkpg@{\hbox{#1}}%
}
\make@find@in{, }
\def\hi@toa@brkpg@#1#2{%
    \hbox{#1,}%
    % Pull the page number to the right margin
    \nobreak\hskip-\rightskip
    % Now permit a line break. If the break is not accepted, nix the
    % right-margin glue above, insert a -1fil glue to counteract the |\hfil| to
    % come, and insert a space to separate the commas.
    \penalty\@tempcnta\hskip \@tempskipa\relax
    % Insert an |\hbox| and then an |\hfil|. If the line is broken, then the
    % |\hfil| will show up, pushing this page number to the right side of the
    % page. If the line is not broken here, then the -1fil glue from above will
    % counteract this glue, and nothing will result.
    \hbox{ }\nobreak\hskip\@tempskipb\relax
    % Reduce the penalty, to encourage later line breaks rather than earlier
    % ones
    \advance\@tempcnta\m@ne
    \find@in{, }{#2}\hi@toa@brkpg@{\hbox{#2}}%
}
```

Citation macros should use this when displaying an author name as the first item in a reference. It stores the author name in `\@this@toatitle`, for identifying duplicates. It then displays the author name, replacing it with `\toaditto` if it is a duplicate. #1 is the font for the author; #2 is the author name (sortable).

```
\DeclareRobustCommand\hi@toa@dupauthor[2]{%
    \if@hi@in@toa
        \global\def\@this@toatitle{#2}%
        \hi@nocap
        \ifx\@this@toatitle\@last@toatitle
            \toaditto, % space
        \else
            #1{#2}, % space
        \fi
    \else
        #1{#2}, % space
    \fi
}
```

If there is no author, then just reset the author tracking state.

```
\DeclareRobustCommand\hi@toa@dupnone{\global\let\@this@toatitle\relax}%
```

For statutes, an even more complex format for Table of Authorities listings is desirable involving ditto marks: the ditto-omitted title should be indented so that it lines up with the non-omitted title.

#1 is a statute title; #2 is a statute citation; #3 is the page number text if no ditto is used; #4 is the text if a ditto is used.

The form of the resulting TOA entry is:

- If no ditto: #1#2#3
- If #1 is empty: [ditto]#4
- If #1 is present: [extra indent][ditto]#4

```
\def\hi@toa@duptitle#1#2#3#4{%
    \global\def\@this@toatitle{#1#2}%
    \ifx\@this@toatitle\@last@toatitle
        \ifhmode \immediate\write16{BAD!!!!}\fi
        \penalty\z@
        \@tempdima=\pagegoal \advance\@tempdima-\pagetotal
        \advance\@tempdima-\pageshrink
        \ifdim\@tempdima<\baselineskip
            \ifvmode \break \fi
            \hi@citeguts{#1#2#3}%
        \else
            \setbox\@tempboxa=\hbox{#2}%
            \@tempdima=\wd\@tempboxa
            \ifstrempty{#1}{}{\advance\@tempdima\toahangindentlen\relax}%
            \ifdim 6em<\@tempdima \@tempdima=6em\relax\fi
            \setbox\@tempboxa=\hbox{\toaditto}%
            \ifdim\@tempdima<\wd\@tempboxa \@tempdima=\wd\@tempboxa \fi
            \hi@citeguts{%
                \leavevmode\hb@xt@\@tempdima{\hss \box\@tempboxa}%
                \hi@nocap
                #4%
            }%
        \fi
    \else
        \hi@citeguts{#1#2#3}%
    \fi
}
```

The following affect the appearance of the Table of Authorities:

- `\toaskip` is a glue specification for how much space to put between entries.
- `passimnum` is a counter for how many items an aggregate page listing must have before it is replaced with *passim*.
- `\PassimText` is the text to be used if `passimnum` is met or exceeded.
- `\toaditto` is the text that replaces repeated authors, statutory titles, and other information that can be omitted from the table listings.
- `\toahangindentlen` is defined as a length for the hanging indent. (It is specified in `em` units so it is defined as macro text.)

```
\newskip\toaskip
\toaskip=1\baselineskip \@plus .1\baselineskip \@minus .1\baselineskip
\newcounter{passimnum} \c@passimnum=\z@
\def\PassimText{\emph{passim}}
\def\toaditto{---\kern-.1em---\kern-.1em---}
\def\toahangindentlen{1.5em}
```

7 Table of Abbreviations

A table of abbreviations is like a simplified table of authorities, and because of *Hereinafter*'s support for abbreviations as the reference type abbrev, such tables can be made automatically. The macro `\tableofabbreviations` takes as an argument a two-element callback, which it executes for every abbreviation definition and full text. Entries are sorted by abbreviations, ignoring any "The".

```
\def\tableofabbreviations#1{%
    \hi@setuptoa
    \@hi@in@toatrue
    \noid
    \IfKnownList{hi@toa@abbrev}{\IfEmptyList{hi@toa@abbrev}{}{%
        \ShowList{hi@toa@abbrev}{%
            \@expand{\@unbrace#1}{\csname lc@##1\endcsname}{iii}%
        }%
    }}{}%
    \@hi@in@toafalse
}
```