nameauth — Name authority mechanism for consistency in text and index*

Charles P. Schaum†

Released 2015/11/11

Abstract

The nameauth package automates the formatting and indexing of names. This aids the use of a **name authority** and the process of textual reordering and revision without needing to retype name references.

Contents

This manual is designed with a text block compatible with both US letter and A4. Macro cross refs are minimized to give a relatively "clean" index that illustrates this package's indexing functions.

*This file describes version v2.0, last revised 2015/11/11.

†E-mail: charles dot schaum at comcast dot net

1 Introduction

1.1 Preliminaries

Books can reference hundreds of names. It takes time and money to check them. This package helps to format and index names consistently and automatically, helping you save time and money. Features include:

- Simultaneously format, display, and index names using fewest keystrokes.
- Context automatically determines the syntactic form of names.
- The default uses longer name forms for the first use and shorter forms thereafter.
- Apply typesetting formats to fit your needs without retyping names.
- Move text in the document without retyping names.
- Show name variants in the text, yet index consistent name forms.
- Handle some cross-cultural naming conventions.
- Allow for different capitalizing and other conventions.
- Index different people with the same name by using tags.
- Process names in list environments and other special environments.

I started using LaTeX and wrote this package when translating old German and Latin texts. I learned much more than I expected regarding LaTeX typesetting engines, font systems, indexing programs, class and package interactions, and naming conventions. Please consider the following general notes that apply to the development and use of this package:

> As of version 2.0 you *can* put control sequences in names, but with a few caveats; see Section 2.5.7. Also, this package is more fault-tolerant and prevents or warns about malformed input.
>
> This manual performs something of a "torture test" on this package. You might want to avoid doing that.
>
> This package depends on etoolbox, suffix, trimspaces, and xargs. It has been tested with `latex`, `lualatex`, `pdflatex`, and `xelatex`. It will work with `makeindex` and `texindy`. This documentation was typeset with `pdflatex` and `makeindex`.
>
> Indexing generally conforms to the standard in Nancy C. Mulvany, *Indexing Books* (Chicago: University of Chicago Press, 1994). This should be suitable for a very wide application across a number of disciplines.

Thanks to Marc van Dongen, Enrico Gregorio, Philipp Stephani, Heiko Oberdiek, Uwe Lueck, and Robert Schlicht for their assistance in the early versions of this package.

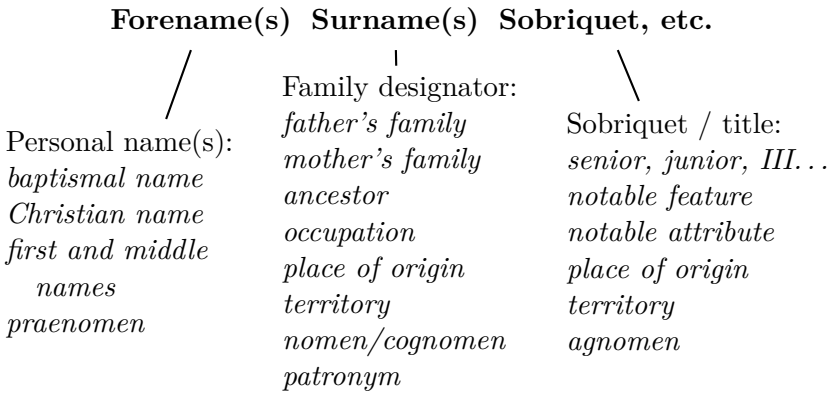
This documentation uses names of living and historical figures because users refer to real people in their projects. At no time do I intending any statement of bias for or against a particular person, culture, or tradition. All names mentioned herein deserve respect for the impact and legacy of their bearers.

1.2 What's In A Name?

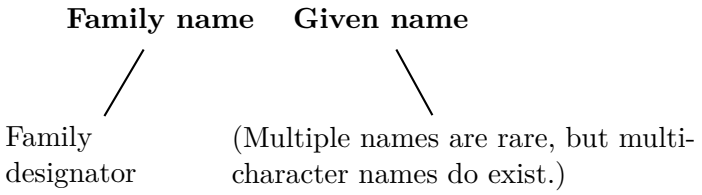
Personal name forms are ambiguous apart from historical and cultural contexts. The nameauth package helps you encode names in a way that anticipates and responds automatically to such contexts without the need to retype names.

Below we see three categories of names with suggestions regarding what might be used therein.[1] Professional writing often calls for the full form of a person's name to be used in its first occurrence, with shorter forms used thereafter. This package adapts that principle to all the forms below.

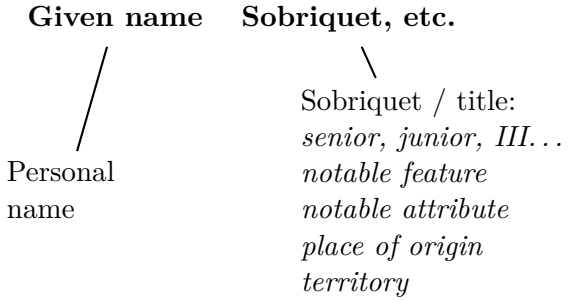
1. Western name:



2. Eastern name:



3. Ancient name:



[1] These suggestions are not exhaustive. For special cases, one might have to decide how to handle, for example, Hungarian and Icelandic names as Eastern or Western.

2 Usage

2.1 Package Options

`\usepackage[`⟨*option*$_1$⟩`,`⟨*option*$_2$⟩`,...]{nameauth}`

Package options address the following:

1. The way name data are entered
2. Semantic display of names (full/short, commas/none, reversed/caps)
3. Typographic display of names (formatted or not)
4. Indexing and sorting of names in the index

Default options are in boldface.

Show/Hide Affix Commas

`nocomma`	**Suppress commas between surnames and affixes, following the *Chicago Manual of Style* and other conventions.**
`comma`	Retain commas between surnames and affixes.

This option is set at load time. If you use *modern standards* or Eastern names, choose the default `nocomma` option to get, *e.g.*, James Earl Carter Jr.

If you need to adopt *older standards* that use commas between surnames and affixes, you have two choices:

- First, the `comma` option produces, *e.g.*, James Earl Carter, Jr. Yet it limits the use of macros like `\AKA` and `\PName`.[2]
- Second, Section 2.4.1 shows how one can use `\ShowComma` with the `nocomma` option to get similar results, but with more flexibility.

Enable/Disable Formatting

`mainmatter`	**Enable typographic formatting attributes (see formatting options below), starting at the beginning of a document.**
`frontmatter`	Disable typographic formatting, but retain automatic full and short forms.

The default starts formatting names immediately. Use the `frontmatter` option to suppress name formatting until you want it to start via `\NamesActive`. These options have no additional effects on the index. See Section 2.5.10.

Enable/Disable Indexing

`index`	**Create index entries in place with names.**
`noindex`	Suppress indexing of names.

The default starts indexing names right away. The `noindex` option disables the indexing of names until `\IndexActive` enables it. This applies only to naming and indexing macros in the nameauth package. See Section 2.8.5.

[2] Before version 0.9 the nameauth package assumed the `comma` option.

Enable/Disable Index Sorting

`pretag`	**Create sort keys used with `makeindex`.**
`nopretag`	Do not create sort keys.

The default allows `\PretagName` to create sort keys used in `makeindex` / `texindy`. Seldom would one change this option. See Section 2.5.8.

Capitalize Entire Surnames

`normalcaps`	**Do not perform any special capitalization.**
`allcaps`	Capitalize entire surnames, such as romanized Eastern names.

This only affects names printed in the body text. To get caps in both the body text and the index, the user should type in the caps manually — an easy task with the simplified interface. Section 2.4.2 details macros that enable capitalization on a section-level and per-name basis.

Reverse Name Order

`notreversed`	**Print names in the order specified by `\Name` and the other macros.**
`allreversed`	Print name forms in "smart" reverse order.
`allrevcomma`	Print all names in Western "Surname, Forenames" order.

See Sections 2.4.1, 2.4.2, and 2.5.1 for related macros to control name reversing by section or by name. So-called "last-comma-first" lists of names are *not* the same as the `comma` option.

Formatting Attributes

`alwaysformat`	If formatting is enabled by the `mainmatter` option or by `\NamesActive`, this option causes all names to have special typographic formatting.
`smallcaps`	**Set the first use of a name in small caps.**
`italic`	Italicize the first occurrence of a name.
`boldface`	Set the first use of a name in boldface.
`noformat`	Do not define a default format. Can be used with custom formatting.

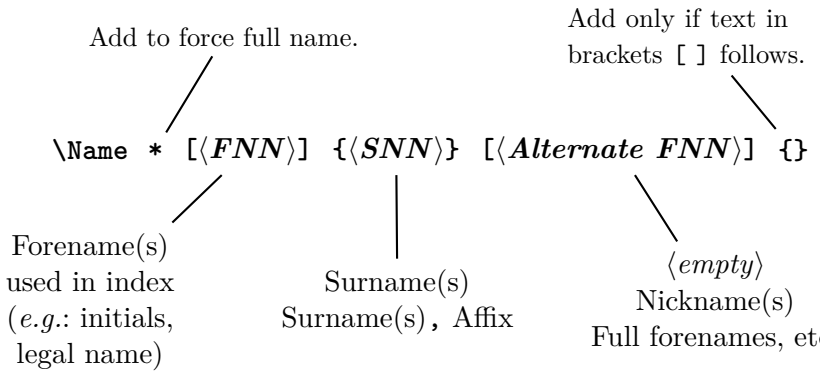
The default is `smallcaps` because this package was developed to aid the editing and translation of older German and Latin documents. Section 2.5.9 details even more possibilities for presenting the first use of names.

2.2 Quick Start Guide

2.2.1 Main Interface

See Section 2.3 for a proper description of `\Name`. Here we see briefly how to work with the classes of names in Section 1.2. We abbreviate the command parameters ⟨*forename(s)*⟩ with ⟨*FNN*⟩ and ⟨*surname(s)*⟩ with ⟨*SNN*⟩. The `nocomma` option works best with affixes, Eastern, and ancient names.

Western Names



Usual forms:

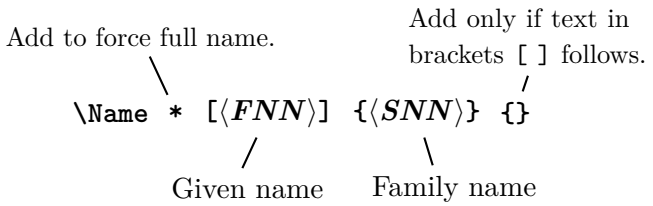
```
\Name[⟨FNN⟩]{⟨SNN⟩}                    \Name[George]{Washington}
\Name[⟨FNN⟩]{⟨SNN, Affix⟩}   \Name[John David]{Rockefeller, II}
```

You must include the ⟨*FNN*⟩ field with alternate forenames. The ⟨*Alternate FNN*⟩ are swapped with the ⟨*FNN*⟩, but only in the body text:

```
\Name[⟨FNN⟩]{⟨SNN⟩}[⟨Alternate FNN⟩]
                                  \Name[Bob]{Hope}[Leslie Townes]
\Name[⟨FNN⟩]{⟨SNN, Affix⟩}[⟨Alternate FNN⟩]
                              \Name[John David]{Rockefeller, IV}[Jay]
```

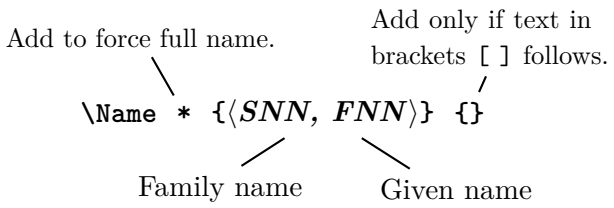
The older syntax is `\Name{⟨SNN⟩}[⟨Affix⟩]`. See Section 2.2.3 for its usage and its shortcomings. It remains for backward compatibility.

Eastern Names in the Text, Western-style Index



Technically, these are Western name forms without affixes. The reversing macros (Section 2.4.2) cause them to display in Eastern order in the body text only. The index entries are Western in fashion: ⟨*SNN*⟩, ⟨*FNN*⟩. This "non-native" form of Eastern names excludes all comma-delimited forms and the old syntax.

Eastern Names in the Text, Eastern-style Index



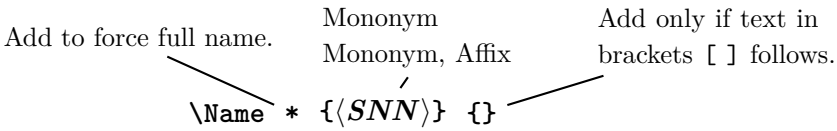
Usual form:

`\Name{`⟨*SNN, FNN*⟩`}` `\Name{Yamamoto, Isoroku}`

These names truly are Eastern names. They will take the form ⟨*SNN FNN*⟩ in the index even if the reversing macros (Section 2.4.2) put the names in Western order in the body text. We call this the "native" form of Eastern names.

The old form of Eastern names is `\Name{`⟨*SNN*⟩`}[`⟨*FNN*⟩`]`. Again, this is retained only for backward compatibility. Cf. Section 2.2.3.

Ancient Names



Usual form:

`\Name{`⟨*SNN*⟩`}` `\Name{Aristotle}`
`\Name{`⟨*SNN, affix*⟩`}` `\Name{Elizabeth, I}`

These forms are used for royalty, ancient figures, and other monomyms.[3] The older syntax is `\Name{`⟨*Mononym*⟩`}[`⟨*Affix*⟩`]`. Cf. Section 2.2.3.

2.2.2 Simplefied Interface

`nameauth` The `nameauth` environment allows one to save typing and aid consistency by using shorthands. It produces control sequences that are fully compatible with the main interface. Although not required, `nameauth` is best used in the document preamble to avoid undefined control sequences.[4]

```
\begin{nameauth}
  \< ⟨cseq1⟩ & ⟨arg1⟩ & ⟨arg2⟩ & ⟨arg3⟩ >
  \< ⟨cseq2⟩ & ⟨arg1⟩ & ⟨arg2⟩ & ⟨arg3⟩ > ...
\end{nameauth}
```

Please bear in mind that `\<` in this context is a control sequence. If you forget it or miss the backslash you will get errors. Leading and trailing spaces in each `&`-delimited field are stripped, as is also the case in the main interface. There *must* be four argument fields (three ampersands) per line.

[3] Technically, the native Eastern forms and the ⟨*Mononym, Affix*⟩ forms are identical, although used in different contexts. You would not wish to reverse a royal name, for example.

[4] The `nameauth` environment uses `\ignorespaces` to mitigate the need for trailing `%`.

Each ⟨*cseq*⟩ creates three macros. In the document text, \⟨*cseq*⟩ itself calls `\Name`. `\L`⟨*cseq*⟩ (think "Long") calls `\Name*`. `\S`⟨*cseq*⟩ (think "Short") calls `\FName`. In the document text, as with `\Name`, include trailing braces `{}` if text in brackets `[]` follows any of the shorthands.

The following example illustrates a fairly complete set of names:

```
\begin{nameauth}
  \< Wash & George & Washington & >                  Western
  \< Soto & Hernando & de Soto & >                   particle
  \< JRock & John David & Rockefeller, II & >         affix
  \< JayR & John David & Rockefeller, IV & Jay >     nickname[5]
  \< Aris & & Aristotle & >                          ancient
  \< Eliz & & Elizabeth, I & >                       royal
  \< Atil & & Atilla, the Hun & >                    ancient
  \< Konoe & Fumimaro & Konoe & >                    Eastern/Western index
  \< Yamt & & Yamamoto, Isoroku & >                  Eastern/Eastern index
\end{nameauth}
```

The following control sequences in the body text illustrate how this environment works. Please pay attention to first and subsequent uses, as well as how the L and S versions and other control sequences modify names:

Basic Uses:

`\Wash`	George Washington
`\LWash`	George Washington
`\Wash`	Washington
`\SWash`	George

Western Reversed with Comma:

`\RevComma\LWash`	Washington, George

Particles:

`\Soto`	Hernando de Soto
`\Soto`	de Soto
`\CapThis\Soto`	De Soto

Affixes:

`\JRock`	John David Rockefeller II
`\LJRock`	John David Rockefeller II
`\JRock`	Rockefeller

Nicknames: (See Section 2.3.2)

`\JayR`	Jay Rockefeller IV
`\SJayR\ \JayR`	Jay Rockefeller

Ancient:

`\Aris`	Aristotle
`\Aris`	Aristotle

Medieval/Royal:

`\Eliz`	Elizabeth I
`\Eliz`	Elizabeth
`\Atil`	Atilla the Hun
`\Atil`	Atilla

Western / Western Index:

`\Konoe`	Fumimaro Konoe
`\LKonoe`	Fumimaro Konoe
`\Konoe`	Konoe

Eastern / Western Index:

`\CapName\RevName\LKonoe`	†KONOE Fumimaro
`\CapName\Konoe`	†KONOE

Eastern / Eastern Index:

`\CapName\Yamt`	YAMAMOTO Isoroku
`\CapName\LYamt`	YAMAMOTO Isoroku
`\CapName\Yamt`	YAMAMOTO

Western / Eastern Index:

`\RevName\LYamt`	Isoroku Yamamoto
`\Yamt`	Yamamoto

In this manual we use a dagger (†) to indicate all Eastern names with Western forms in the index. The use is encouraged to compare the index entries with the body text in order to understand better what the macros are doing.

[5] `\SJayR` always gives "Jay." See Section 2.3.2 for possible difficulties when using this variation. Use `\AKA` to create a *see* reference from Jay Rockefeller to his full index entry: "Rockefeller, John David, IV." For more on `\AKA` see Section 2.7.1.

2.2.3 Older Syntax

An "obsolete" syntax remains for backward compatibility with early versions of nameauth and with the comma option. Please avoid mixing the older and newer forms. The old form lacks some error checking and robustness contained in the new syntax and limits the use of several macros:

```
\Name{Dagobert}[I]                                  royal name
\Name{Yoshida}[Shigeru]                             Eastern name
\begin{nameauth}
  \< Dagb & & Dagobert & I >                        royal name
  \< Yosh & & Yoshida & Shigeru >                   Eastern name
\end{nameauth}
```

\Dagb gives DAGOBERT I, while the next \Dagb produces Dagobert. In similar fashion as before, we see \LDagb Dagobert I, \CapName\Yosh YOSHIDA SHIGERU, and \CapName\RevName\LYosh Shigeru YOSHIDA.

- In the old syntax, \Name{Henry}[VIII] prints "HENRY VIII" and "Henry." If you mix \Name{Henry}[VIII] with the newer \Name{Henry, VIII} they both print HENRY VIII and Henry in the body text. Yet they generate different control sequences for both first/subsequent uses and index tags.[6]
- Erroneously typing \Name[Henry]{VIII} prints "HENRY VIII" and "VIII," as well as producing a malformed index entry. As of version 2.0, this kind of malformed input creates no side effects for well-formed input.
- Avoid forms like \Name[Henry]{VIII}[Tudor] which gives "Tudor VIII" and "VIII." Again, this classifies as malformed input.
- Also avoid \Name{Henry, VIII}[Tudor] unless you really want "HENRY VIII TUDOR" and "Henry" in the body text and "Henry VIII" in the index.
- One solution to incorporate the house designation adds "Tudor" as needed in the text to \Name{Henry, VIII} and uses a tag in the index: \TagName{Henry, VIII}{ Tudor} (see Section 2.8.2).
- The older syntax will not work with some macros. From the film *Men in Black III*, \AKA{Boris}[the Animal]{Just Boris} fails. \PName fails for the same reasons. See also Section 2.7.1
- This form does work: \Name{Boris, the Animal} \AKA{Boris, the Animal}{Just Boris}. You get BORIS THE ANIMAL being "Just Boris."

[6] Technically you can mix the two, as I do here. This package is all about offering choices, not restricting authors. Yet you must force first and subsequent uses with \ForgetName and \SubvertName, and make common index tags, *e.g.*: \TagName{Henry, VIII}{, king} and \TagName{Henry}[VIII]{, king}. That undermines the time-saving features offered by this package. Since authors have personal styles and workflows, this package tries its best to embrace the horror of inherent ambiguity in names and their usage.

2.2.4 Tips and Warnings

- Keep it simple! Avoid unneeded macros and use the simplified interface.
- Check braces and brackets with naming macros to avoid errors like "Paragraph ended. . . " and "Missing ⟨*grouping token*⟩ inserted."
- For stage names, etc., try using *e.g.*: `\Name[J.]{Kreskin}[The Amazing]` `(\AKA[J.]{Kreskin}[Joseph]{Kresge})`. As a result you get The Amazing Kreskin (Joseph Kresge). The corresponding index entry will be "Kreskin, J." You must have a name in the first optional field for this to work.
- Let this package help your workflow as you need it, instead of trying to figure out all its intricacies. Special cases like "Iron Mike" Tyson as the nickname for Mike Tyson may be handled in a number of ways.
  1. Follow ``` ``Iron Mike'' ``` with `\IndexName[Mike]{Tyson}` and do whatever you want in the text.
  2. `\SubvertName[Mike]{Tyson}\FName[Mike]{Tyson}[Iron Mike]%` `\Name[Mike]{Tyson}` gives "Iron Mike Tyson." You are responsible for typesetting the first use and you can `\let` the whole thing to a control sequence.
  3. Use ``` ``\AKA[Mike]{Tyson}{Iron Mike}'' ``` to create "Iron Mike" in the text and a *see*-type cross-reference to "Tyson, Mike" in the index. Be sure to have an occurrence of `\Name[Mike]{Tyson}` in the text. See also Section 2.7.1.
- Omit spaces between initials if possible; see also Bringhurst's *Elements of Typographic Style*. If your publisher wants spaces between initials, use `\frenchspacing`[7] or put thin spaces `\,` between them. Consider:

  ```
  \begin{nameauth}
    \< White & E.\,B. & White & >
  \end{nameauth}
  ```

  Compare the following:

`\White` and `\LWhite`	(E. B. White and E. B. White)
Normal:	E. B. White and E. B. White.

- One way to spot errors is to compare index entries with names in the body text. All macros that produce output also emit meaningful warnings. `\PName` produces warnings via `\Name` and `\AKA`.
- Please pay greater attention to the warnings produced by `\IndexName`, `\TagName`, `\UntagName`, and `\ExcludeName`. Many other warnings are FYI.
- The `nameauth` environment only generates a warning when you overwrite a control sequence designator with a new name.
- Inserting an empty string in a required field will generate an error.

[7] Although there should be no difference between `\frenchspacing` and `\nonfrenchspacing` when it comes to initials, some classes or packages may affect that.

Warnings result from:

- Using a cross-reference `[`⟨*alternate FNN*⟩`]{`⟨*alternate SNN*⟩`}[`⟨*alt. names*⟩`]` created by `\AKA` as a name reference in `\Name`, `\FName`, and `\PName`. They merely will print a name in the body text.
- Using a name reference `[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[`⟨*Alternate names*⟩`]` created by `\Name`, `\FName`, and `\PName` as a cross-reference in `\AKA`. It merely will print a name in the body text.
- Using `\AKA` to create the same cross-reference multiple times or with a cross-reference created by `\ExcludeName`. It merely will print a name in the body text, but not the index.
- Using `\IndexName` to index a cross-reference made via `\AKA` or via the mechanism in `\ExcludeName` as a main entry. It will do nothing.
- Using `\TagName`, `\UntagName`, and `\PretagName` with cross-references. The first two will do nothing. However, `\PretagName` will "pretag" a cross-reference. This is the desired behavior.
- Using `\ExcludeName` with cross-references. It will do nothing.
- Using `\ExcludeName` to exclude a name that has already been used. Likewise, it will do nothing.
- Using `\Name`, `\FName`, `\PName`, and `\AKA` to refer to names and cross-references excluded by `\ExcludeName`. They merely will print a name in the body text.
- Using the `nameauth` environment to redefine shorthands, such as:

```
\begin{nameauth}
  \< White & E.\,B. & White & >
  \< White & E. B. & White & >
\end{nameauth}
```

2.3 Naming Macros

2.3.1 \Name and \Name*

\Name
\Name*

This macro generates two forms of the name: a printed form in the text and a form of the name that occurs in the index. The general syntax is:

\Name[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]
\Name*[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]

Here we see how the syntax works:

\Name[Albert]{Einstein}	Albert Einstein
\Name*[Albert]{Einstein}	Albert Einstein
\Name[Albert]{Einstein}	Einstein
\Name{Confucius}	Confucius
\Name*{Confucius}	Confucius
\Name{Confucius}	Confucius
\Name[M.T.]{Cicero}[Marcus Tullius]	Marcus Tullius Cicero
\Name*[M.T.]{Cicero}[Marcus Tullius]	Marcus Tullius Cicero
\Name[M.T.]{Cicero}[Marcus Tullius]	Cicero
\Name{Charles, the Bald}	Charles the Bald
\Name*{Charles, the Bald}	Charles the Bald
\Name{Charles, the Bald}	Charles

\Name displays and indexes names, as illustrated in Section 2.10. It always prints the ⟨*SNN*⟩ field. \Name prints the "full name" at the first occurrence, then the partial form thereafter. \Name* always prints the full name.

The ⟨*Alternate names*⟩ field replaces the ⟨*FNN*⟩ field in the text only if the ⟨*FNN*⟩ field is not empty; see "Cicero" above. One can use a nickname in some instances while keeping the indexed form constant.

Thus, regarding their index entries and first/subsequent uses in nameauth, \Name[M.T.]{Cicero}[Marcus Tullius] and \Name[M.T.]{Cicero} are equivalent, while \Name{Cicero}[Marcus Tullius] and \Name{Cicero} are not. Generally avoid older forms like \Name{Charles}[the Bald].

```
\begin{nameauth}
  \< Einstein & Albert & Einstein & >
  \< Cicero & M.T. & Cicero & >
  \< Confucius & & Confucius & >
  \< CBald & & Charles, the Bald & >
\end{nameauth}
```

Above we see the same setup with the simplified interface. In the body text, \Einstein, \LEinstein, and \Einstein produce Albert Einstein, Albert Einstein, and Einstein. \CBald\ and \CBald give Charles the Bald and Charles. The next section demonstrates that \LCicero[Marcus Tullius] allows for M.T. Cicero to be both Marcus Tullius Cicero and M.T. Cicero.

2.3.2 Forenames: \FName

\FName
\FName*

This casual friend of \Name prints only "first" names, but it will still print a full name when a first use occurs. \FName is intended for Western-style names. \FName* is only a synonym for \FName. The syntax is basically the same:

\FName[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]

The first reference to \FName always is a full name. That prevents a first-name-only reference before a person has been introduced. Intentionally, \FName *never* gives the first name with Eastern names. For examples we see below:

\FName[Albert]{Einstein}	Albert Einstein
\FName[Albert]{Einstein}	Albert
\FName{Confucius}	Confucius
\FName{Confucius}	Confucius
\FName[M.T.]{Cicero}[Marcus Tullius]	Marcus Tullius Cicero
\FName[M.T.]{Cicero}[Marcus Tullius]	Marcus Tullius
\FName{Charles, the Bald}	Charles the Bald
\FName{Charles, the Bald}	Charles

The Cicero example shows how the ⟨*Alternate names*⟩ field can work. Be careful with nicknames: ``\FName[Chesley B.]{Sullenberger, III}[Sully]'' produces "Sully Sullenberger III" and "Sully." This is not a "bug" as such, as the Cicero example illustrates. Names are very context-sensitive.

With \SEinstein, \SConfucius, \SCicero, and \SCBald we get Albert, Confucius, M.T., and Charles when using the simplified interface example from the previous page. \SCicero[Marcus Tullius] gives Marcus Tullius. The simplified interface allows you to use a "default nickname." In Section 2.2.2 \SJayR gives Jay Rockefeller IV and Jay, but the index entry "Rockefeller, John David, IV." Yet default nicknames are not always what they seem:

```
\begin{nameauth}
  \< Ches & Chesley B. & Sullenberger, III & >
  \< Sully & Chesley B. & Sullenberger, III & Sully >
\end{nameauth}
```

The first use \Ches prints "Chesley B. Sullenberger III." Later, \SChes and \SSully print "Chesley B." and "Sully." While \SChes[Sully] always gives, "Sully," \SSully[Chesley B.] prints "Sully[Chesley B.]."

\SSully[Chesley B.] expands to what it should be, not what you might expect it to be:\FName[Chesley B.]{Sullenberger, III}[Sully][Chesley B.]. Thus we prefer \LCicero[Marcus Tullius] and \SCicero[Marcus Tullius]:

```
\begin{nameauth}
  \< Cicero & M.T. & Cicero & >                    preferred
  \< Cicero & M.T. & Cicero & Marcus Tullius > not preferred
\end{nameauth}
```

This package offers preferred choices, but its design does not force you to use them because names and their uses have many variants.

2.4 Affixes and Eastern Names

2.4.1 Affixes Need Commas

Comma-delimited affixes handle several different name types. *Always include a comma as an affix delimiter*, even when the `nocomma` option does not print the comma. Extra spaces and trailing commas are ignored. Other name types include royal, medieval, and Eastern names:

`\Name[Oskar]{Hammerstein, II}`	Oskar Hammerstein II
`\Name[Oskar]{Hammerstein, II}`	Hammerstein
`\Name{Louis, XIV}`	Louis XIV
`\Name{Louis, XIV}`	Louis
`\Name{Sun, Yat-sen}`	Sun Yat-sen
`\Name{Sun, Yat-sen}`	Sun

One must use comma-delimited suffixes when cross-referencing affixed Western names, royal names, some medieval names, and Eastern names with `\AKA`; see Section 2.7.1.

`\KeepAffix` Put `\KeepAffix` before `\Name` or `\AKA` if a line break or page break divides a ⟨*SNN, affix*⟩ pair. This puts a non-breaking space between *SNN* and *affix* in the body text, but not in the index. Other options to fix bad breaks include using `\hbox`, kerning and spacing in the microtype package, etc.

`\ShowComma` The `comma` option is restrictive and used to reproduce older texts. `\ShowComma` gets the same results on a per-name basis while using the default `nocomma` option. With `\ShowComma\Name[Louis]{Gossett, Jr.}` one gets Louis Gossett, Jr. One must use `\ShowComma` consistently or risk errors in the body text and index.

Avoid using the older syntax, shown below, except with the `comma` option. It does not handle Western names with affixes and some other name types. `\AKA` and `\PName` cannot create cross-references to these forms. These older forms include:

`\Name{Henry}[VIII]`	Henry VIII
`\Name{Henry}[VIII]`	Henry
`\Name{Chiang}[Kai-shek]`	Chiang Kai-shek
`\Name{Chiang}[Kai-shek]`	Chiang

These older forms work because no ⟨*FNN*⟩ are present. Otherwise you would get weird nicknames. Again, please avoid using the older syntax.

2.4.2 Eastern Names

The nameauth package offers "non-native" and "native" ways to handle romanized Eastern names. `\RevName\Name[`⟨*Eastern FNN*⟩`]{`⟨*Eastern SNN*⟩`}` will produce an Eastern name in the body text and the Western form ⟨*SNN*⟩, ⟨*FNN*⟩ in the index, including the comma. We call this "non-native" mode.

In contrast, both `\Name{`⟨*Eastern SNN, Eastern FNN*⟩`}` and the older syntax `\Name{`⟨*Eastern SNN*⟩`}[`⟨*Eastern FNN*⟩`]` produce an Eastern name form in the body text: ⟨*SNN*⟩ ⟨*FNN*⟩ as well as in the index. This form has no comma in the index. We call this "native" mode. Offering these two modes gives the greatest flexibility in indexing requirements.

`\ReverseActive`
`\ReverseInactive`
`\RevName`

The "smart" reverse output mechanism converts between Western and Eastern forms in the text, but not the index. If one wants a Western-format index, then pick non-native mode. If Eastern forms are okay in the index, then pick native mode. In addition to the class options described in Section 2.1, `\ReverseActive` and `\ReverseInactive` toggle reversing on a larger scale, while `\RevName` is used once per `\Name`.

This list of Japanese music artists shows `\RevName` in action. Names in Western order, then non-native Eastern order are marked with a dagger (†). All other names are in native Eastern, then Western order. Forms using the old syntax are in parentheses. Name formatting is turned off in order to focus on reversing:

	unchanged	`\RevName`
`†\Name*[Aiko]{Nakano}`	†Aiko Nakano	†Nakano Aiko
`\Name*{Arai, Akino}`	Arai Akino	Akino Arai
`(\Name*{Ishida}[Yoko])`	(Ishida Yoko)	(Yoko Ishida)
`\Name*{Yohko}`	Yohko	Yohko

`\AllCapsActive`
`\AllCapsInactive`
`\CapName`

Use `\AllCapsActive`, `\AllCapsInactive`, and `\CapName` for fully-capitalized family names in the body text. These macros are analogous to the reversing macros above and may be used alone or with those and other state-toggling macros, *e.g.* `\CapName\RevName\Name`. Names in Western order, then non-native Eastern order are marked with a dagger (†). All other names are in native Eastern, then Western order. Forms using the old syntax are in parentheses. Name formatting is turned off in order to focus on capitalizing and reversing:

	unchanged	`\CapName\RevName`
`†\Name*[Yoko]{Kanno}`	†Yoko KANNO	†KANNO Yoko
`\Name*{Shikata, Akiko}`	SHIKATA Akiko	Akiko SHIKATA
`(\Name*{Nogawa}[Sakura])`	(NOGAWA Sakura)	(Sakura NOGAWA)
`\Name*{Yohko}`	YOHKO	YOHKO

Notice how capitalization is independent of formatting. The reversing and capitalization macros also work with `\AKA`. They affect only the text, not the index. Whoever wants all-cap forms in the index will have to capitalize everything manually or modify the macros.

2.5 Other Naming Topics

2.5.1 Listing by Surname

\ReverseCommaActive
\ReverseCommaInactive
\RevComma

The reversing macros `\ReverseCommaActive`, `\ReverseCommaInactive`, and `\RevComma` allow the easy generation of name lists ordered as ⟨*surname*⟩, ⟨*forename(s)*⟩. The first two are broad toggles, while the third works on a per-name basis. Eastern, medieval, and royal names do not work with these macros. Name formatting has been turned off to focus on reversing and commas:

John Stuart Mill	Mill, John Stuart	OK
Oskar Hammerstein II	Hammerstein II, Oskar	OK
John Eriugena	Eriugena John	incompatible
Mao Tse-tung	Tse-tung Mao	incompatible
Anaximander	Anaximander	OK

2.5.2 Naming Standards

\CapThis

According to the *Chicago Manual of Style*, English names with the particles *de*, *de la*, *d'*, *von*, *van*, and *ten* generally keep them with the last name, using varied capitalization. *Le*, *La*, and *L'* always are capitalized unless preceded by *de*.

In English, these particles go in the ⟨*SNN*⟩ field of `\Name`, *e.g.*, Walter de la Mare. To capitalize *de* when it arises at the beginning of a sentence, use `\CapThis\Name[Walter]{de la Mare}`. De la Mare will think it fair. Non-English contexts do not always bind particles to surnames. Using `\Name` and `\FName` with alternate forenames helps to address this issue of different standards. See also Section 2.10.2.

2.5.3 Fault Tolerance

Especially since version 2.0, the `nameauth` package has been redesigned to help prevent malformed input from generating odd side effects. For example, `\Name[Henry]{VIII}` and `\Name{Henry}[VIII]` used to create the same control sequence and allow one to affect the well-formed input via the malformed input. That no longer happens. Furthermore, we guard against deliberately empty required values being passed to the naming macros.

Perhaps a vexing issue with older versions of the package was that leading and trailing spaces in the input could cause significant and obscure errors. Currently, both leading and trailing spaces — but not medial spaces — are ignored in the naming macros, making the following equivalent:

Macro Example	*Resulting text*
`\Name*[ Martin Luther]{King, Jr.}`	Martin Luther King Jr.
`\Name*[Martin Luther ]{King, Jr.}`	Martin Luther King Jr.
`\Name*[ Martin Luther ]{King, Jr.}`	Martin Luther King Jr.
`\Name*[Martin Luther]{ King, Jr.}`	Martin Luther King Jr.
`\Name*[Martin Luther]{King, Jr. }`	Martin Luther King Jr.
`\Name*[Martin Luther]{ King, Jr. }`	Martin Luther King Jr.

All name input fields of `\Name`, `\FName`, `\AKA`, and `\IndexName` are so protected. The other macros really are not affected by these issues.

2.5.4 Hyphenation

The simplified interface trivializes the consistent insertion of optional hyphens in names, as we see below:

```
\begin{nameauth}
  \< Bier & Johann & Bier\-mann & >
\end{nameauth}
```

produces Johann Biermann and Biermann. This should prevent the break "Biermann," which could happen otherwise. You can even tag and untag such forms. The bad break above was manufactured, while the bad break below is actual.

Bad breaks can be fixed with the babel or polyglossia packages. John Strietelmeier can have a bad break in English, as you see. Using babel, we can use the following example so that `\de{\Name*[John]{Strietelmeier}}` generates John Strietelmeier and helps prevent bad breaks:

```
\newcommand{\de}[1]{\foreignlanguage{ngerman}{#1}}
```

2.5.5 Indexing and babel

texindy

Using babel with Roman page numbers will put `\textlatin` in the index entries if one includes a language that does not use the Latin alphabet — even if the main language does. The `texindy` program will ignore such references. This issue can affect nameauth. One workaround for `texindy` could enclose text with any macros that write to the index in an environment or a `\long` macro defined like:

```
\newcommand{\fix}[1]{\def\textlatin##1{##1}#1}
```

2.5.6 Detecting Punctuation

In Western names, some affixes with full stops could appear at the end of a sentence. Such affixes include "Jr." (junior), "Sr." (senior), "d. J." (*der Jüngere*), and "d. Ä." (*der Ältere*). Consider this example, where some lines have two full stops and some do not:

Macro Example		*Resulting text*
`\Name[Martin Luther]{King, Jr.}.`	$2 \to 1$	Martin Luther King Jr.
`\Name[Martin Luther]{King, Jr.}.`	$2 \to 1$	King.
`\Name[Martin Luther]{King, Jr.}␣`	$1 \to 0$	King
`\Name*[Martin Luther]{King, Jr.}.`	$2 \to 1$	Martin Luther King Jr.
`\Name*[Martin Luther]{King, Jr.}␣`	$1 \to 1$	Martin Luther King Jr.
`{\Name*[Martin Luther]{King, Jr.}}.`	$2 \to 2$	Martin Luther King Jr..[8]

`\Name`, `\FName`, and `\AKA` all check for a trailing full stop in the printed name in the text. If it exists, and if the next token is also a full stop, they gobble the trailing full stop. Grouping tokens, among other items, can frustrate this detection, as shown in the previous example.

[8] Example of how to frustrate the full stop detection mechanism.

2.5.7 Accented Names

For texts that contain significant amounts of accented characters, using `xindy` (`texindy`) and `xelatex` or `lualatex` is recommended.

Under NFSS your results may vary with the `utf8` input encoding and `T1` or `TS1` font encoding. First of all, at the start of a name field, you need to put Unicode characters in braces or use control sequences, *e.g.*, `\Name{{Æ}thelred, II}` and `\Name{\AE thelred, II}`. With `\Name[Johann]{Andreä}` we see that subsequent letters need no changes. For the first letter in any argument field of the nameauth macros, if you fail to put accented letters in braces or use control sequences, you will get an error and execution will halt. You can use these glyphs:

À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ	FIRST USE
À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ	second use
Ò Ó Ô Õ Ö Ø	Ù Ú Û Ü Ý	Þ SS	FIRST USE
Ò Ó Ô Õ Ö Ø	Ù Ú Û Ü Ý	Þ ß	second use
À Á Â Ã Ä Å Æ	Ç È É Ê Ë	Ì Í Î Ï Ð Ñ	FIRST USE
à á â ã ä å æ	ç è é ê ë	ì í î ï ð ñ	second use
ò ó ô õ ö ø	ù ú û ü ý	þ ÿ	FIRST USE
ò ó ô õ ö ø	ù ú û ü ý	þ ÿ	second use
Ă ă Ą ą Ć ć Č č	Ď ď Đ đ Ę ę Ě ě	Ğ ğ İ ı	FIRST USE
Ă ă Ą ą Ć ć Č č	Ď ď Đ đ Ę ę Ě ě	Ğ ğ İ ı	second use
IJ ıj Ľ ŀ Ł ł	Ń ń Ň ň Œ œ	Ŕ ŕ Ř ř	FIRST USE
IJ ij Ľ ľ Ł ł	Ń ń Ň ň Œ œ	Ŕ ŕ Ř ř	second use
Ś Ş ş Š š Ţ ţ Ť ť	Ů ů Ű ű	Ź ź Ż ż Ž ž	FIRST USE
Ś Ş ş Š š Ţ ţ Ť ŧ	Ů ů Ű ű	Ź ź Ż ż Ž ž	second use

The first example below shows that Unicode characters and control sequences are not interchangeable. Notice again that only the first Unicode character need be a control sequence or put in braces:

```
\Name[Johann]{Andre\"a}
\Name[Johann]{Andre\"a}
\Name[Johann]{Andreä}
\Name[Johann]{Andreä}

\Name{\AE thelred, II}
\Name{\AE thelred, II}
\Name{{Æ}thelred, II}
\Name{{Æ}thelred, II}
```

JOHANN ANDREÄ
Andreä
JOHANN ANDREÄ
Andreä

ÆTHELRED II
Æthelred
ÆTHELRED II
Æthelred

Additional accents and glyphs can be used with Unicode input, NFSS, inputenc, and fontenc when using fonts with TS1 glyphs, *e.g.*, `\usepackage{lmodern}` (per the table on pages 455–63 in *The Latex Companion*):

```
\usepackage{newunicodechar}
\DeclareTextSymbolDefault{\textlongs}{TS1}
\DeclareTextSymbol{\textlongs}{TS1}{115}
\newunicodechar{ſ}{\textlongs}
```

That lets you type "In Congreſs, July 4, 1776." `\newunicodechar{ā}{\=a}` allows `\Name{Ghazāli}` to generate GHAZĀLI.

Control sequences like `\=a` fail when using `makeindex` and `gind.ist`, such as with the ltxdoc class, because the equal sign is an "actual" character instead of `@`. Using `\index{Gh{\=a}zali}` halts execution. Using `\index{Gh\=azali}` gives an "azali" entry sorted under "Gh" (thanks Dan Luecking). This issue is not specific to nameauth and affects any document where one uses `gind.ist`.

One may use expandable control sequences in names (thanks Robert Schlicht). Also, you can define letters with `\edef` and `\noexpand` to use in names, as some do to "protect" accented letters in names. As of version 2.0 of nameauth helpful concerns expressed by Patrick Cousot have been addressed.

2.5.8 Index Sorting

The general practice for sorting with `makeindex -g` involves creating your own `.ist` file (pages 659–65 in *The Latex Companion*). Otherwise use the following form that works with both `makeindex` and `texindy`:

`\index{`⟨*sortkey*⟩`@`⟨*actual*⟩`}`

Before version 2.0 of nameauth, one had to sort and index a name like Jan Łukasiewicz by putting it between `\IndexInactive` and `\IndexActive` while creating a manual index entry.

`\PretagName`

Fortunately, the current versions of nameauth have adopted an easier solution. The syntax of `\PretagName` is like that of `\TagName`:

`\PretagName[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[`⟨*Alternate names*⟩`]{`⟨*tag*⟩`}`

The `\PretagName` macro does not work exactly like the `\TagName` and `\UntagName` macros (see Section 2.8.2 and following). The main differences are:

- You can "pretag" any name and any cross-reference.
- You can "tag" and "untag" only names, not cross-references.
- There is no command to undo a "pretag."

The `\PretagName` macro creates a sort key terminated with the "actual" character, which is `@` by default. Do not include the "actual" character in the pretag. Now, sorting index entries is as simple as:

```
\PretagName[Jan]{{Ł}ukasiewicz}{Lukasiewicz, Jan}
\PretagName{{Æ}thelred, II}{Aethelred 2}
```

One need only pretag names once in the preamble. Every time that Łukasiewicz or Æthelred are referenced, the proper index entry will be created. If you create a cross-reference with `\AKA` and you want to pretag it, see Section 2.7.1.

`\IndexActual`

If you need to change the "actual" character, such as with `gind.ist`, put `\IndexActual{=}` in the preamble.

> You cannot use index tags if the nameauth indexing feature is inactive.

This package tries to work with multiple languages and typesetting engines. The following preamble snippet illustrates how that can be done:

```
\usepackage{ifxetex}
\usepackage{ifluatex}
\ifxetex                                          % uses fontspec
  \usepackage{fontspec}
  \defaultfontfeatures{Mapping=tex-text}
  \usepackage{xunicode}
  \usepackage{xltxtra}
\else
  \ifluatex                                  % also uses fontspec
    \usepackage{fontspec}
    \defaultfontfeatures{Ligatures=TeX}
  \else                                        % traditional NFSS
    \usepackage[utf8]{inputenc}
    \usepackage[TS1,T1]{fontenc}
  \fi
\fi
```

The following can be used in the text itself to allow for conditional processing that helps one to document work under multiple engines:

```
\ifxetex ⟨xelatex text⟩%
\else
  \ifluatex
    \ifpdf ⟨lualatex in pdf mode text⟩%
    \else ⟨lualatex in dvi mode text⟩%
    \fi
  \else
    \ifpdf ⟨pdflatex text⟩%
    \else ⟨latex text⟩%
    \fi
  \fi
\fi
```

2.5.9 Custom Formatting

`\NamesFormat` When formatting is active, `\NamesFormat` is called at the first instance of a name, and at every instance of a name when the `alwaysformat` option is used. Beyond using the package options, one also can redefine `\NamesFormat` to create custom effects. For example, one might change or suppress formatting in all footnotes:

```
\makeatletter
\let\@oldfntext\@makefntext
\long\def\@makefntext#1{\def\NamesFormat{}\@oldfntext{#1}}
\makeatother
```

This approach will not print the first use of a name in the body text if it already occured in the footnotes unless one uses `\ForgetName` to force that. This example takes advantage of the deep scoping of `\@makefntext` in order to use a localized `\def` to make a temporary change. The next section shows how one can use a completely independent system of first and subsequent use in the footnotes.

A second example puts the mention of first names in boldface, with additional notations in the margin if possible:

```
\let\oldformat\NamesFormat
\renewcommand{\NamesFormat}[1]%
  {\textbf{#1}\ifinner\else
  \marginpar{\raggedleft\scriptsize #1}\fi}
\PretagName{Vlad, Ţepeş}{Vlad Tepes}%
```

```
\Name{Vlad III, Dracula}, known as \AKA{Vlad III, Dracula}{Vlad, Ţepeş},
``\AKA*{Vlad III, Dracula}{Vlad}[the Impaler]'' after his death, was the
son of \Name{Vlad II, Dracul}, a member of the Order of the Dragon. Later
references to ``\Name{Vlad III, Dracula}'' appear thus.
```

Vlad III Dracula
Vlad II Dracul

Vlad III Dracula, known as Vlad Ţepeş, "the Impaler" after his death, was the son of **Vlad II Dracul**, a member of the Order of the Dragon. Later references to "Vlad III" appear thus.

The `quote` environment used above permits local changes to `\NamesFormat` so they revert back to the default: Vlad III Dracula and Vlad III. For references to "Vlad" instead of "Vlad III" one could use `\Name{Vlad, III Dracula}`.[9]

2.5.10 Disable Formatting

`\NamesActive`
`\NamesInactive`

Using the `frontmatter` option deactivates formatting until `\NamesActive` occurs. Another macro, `\NamesInactive`, will deactivate formatting again. These two macros toggle two independent systems of formatting and first use.

Here we switch to the "front matter" mode with `\NamesInactive`:

`\Name[Rudolph]{Carnap}`	Rudolph Carnap
`\Name[Rudolph]{Carnap}`	Carnap
`\Name[Nicolas]{Malebranche}`	Nicolas Malebranche
`\Name[Nicolas]{Malebranche}`	Malebranche

Then we switch back to "main matter" mode with `\NamesActive`:

`\Name[Rudolph]{Carnap}`	Rudolph Carnap
`\Name[Rudolph]{Carnap}`	Carnap
`\Name[Nicolas]{Malebranche}`	Nicolas Malebranche
`\Name[Nicolas]{Malebranche}`	Malebranche

Notice that we have two independent cases of "first use" above. Consider the two "species" of names to be "non-formatted" and "formatted," intended for front matter and main matter. Yet one could use this, *e.g.*, in footnotes:

```
\makeatletter
\let\@oldfntext\@makefntext
\long\def\@makefntext#1{%
  \NamesInactive\@oldfntext{#1}\NamesActive%
}\makeatother
```

[9] Do not mix `\Name{Vlad III, Dracula}` with `\Name{Vlad, III Dracula}` or the old syntax, lest errors bite! You would get multiple index entries with `\Name`, unwanted cross-references with `\AKA` and unexpected forms in the text. The simplified interface helps one to avoid this.

2.6 Tweaks: \ForgetName and \SubvertName

Perhaps the easiest way to avoid the "interspecies clashes" above are the two macros presented here. They are meant for tweaking text at or near final draft stage. They affect both front matter and main matter.

\ForgetName

This macro is a "dirty trick" of sorts that takes the same optional and mandatory parameters used by \Name. It handles its arguments in the same way, except that it ignores the final parameter if ⟨*FNN*⟩ are present. The syntax is:

\ForgetName[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]

This macro causes \Name and friends to "forget" prior uses of a name. The next use of that name will print as if it were a "first use," even if it is not. Index entries and cross-references are *never* forgotten by this package.

\SubvertName

This macro is the opposite of the one above. It takes the same parameters. It handles its arguments in the same manner. The syntax is:

\SubvertName[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]

This macro causes \Name and friends to think that a prior use of a name already has occurred. The next use of that name will print as if it were a "subsequent use," even if it is not.

2.7 Name Variant Macros

2.7.1 \AKA

\AKA
\AKA*

\AKA (meaning *also known as*) handles pseudonyms, stage names, *noms de plume*, and so on in order to replace typing manual cross-references in the index:

```
\Name{Jean, sans Peur} (\AKA{Jean, sans Peur}{Jean the Fearless})
was Duke of Burgundy 1404--1419.
```

Jean sans Peur (Jean the Fearless) was Duke of Burgundy 1404–1419.

Notice that "Jean the Fearless" receives no special formatting. This is intentional, as it reflects the idea of formatting only those names with main index entries. Nevertheless, the reversing and capitalizing mechanisms do work with \AKA. The syntax for \AKA is:

\AKA[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alt. FNN*⟩]{⟨*Alt. SNN*⟩}[⟨*Alt. names*⟩]
\AKA*[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alt. FNN*⟩]{⟨*Alt. SNN*⟩}[⟨*Alt. names*⟩]

Only the ⟨*FNN*⟩ and ⟨*SNN*⟩ arguments from \Name and friends may be cross-referenced. The new syntax allows \AKA to cross-reference all name types. Both macros create a cross-reference in the index from the ⟨*Alt. FNN*⟩, ⟨*Alt. SNN*⟩, and ⟨*Alt. names*⟩ fields to a name defined by ⟨*FNN*⟩ and ⟨*SNN*⟩, regardless of whether that name has been used.

Both macros print only the ⟨*Alt. FNN*⟩ and ⟨*Alt. SNN*⟩ fields in the body text. If the ⟨*Alt. names*⟩ field is present, \AKA swaps ⟨*Alt. names*⟩ with ⟨*Alt. FNN*⟩ in the body text. \AKA* just prints ⟨*Alt. names*⟩ (if present) in the body text. See also Section 2.8.2.

For the following name types, `\AKA` and `\AKA*` yield the same results, using Bob Hope, Louis XIV, and Lao-tzu as examples:

`\AKA[Bob]{Hope}[Leslie Townes]{Hope}`	Leslie Townes Hope
`\AKA{Louis, XIV}{Sun King}`	Sun King
`\AKA{Lao-tzu}{Li, Er}`	Li Er

`\AKA` ignores the ⟨*Alt. names*⟩ field with the names above. The ⟨*Alt. names*⟩ field as part of the cross-reference was envisaged for names like Gregory I:

`\AKA{Gregory, I}{Gregory}[the Great]`	Gregory the Great
`\AKA*{Gregory, I}{Gregory}[the Great]`	the Great

Using `\Name*{Gregory, I} ``\AKA*{Gregory, I}{Gregory}[the Great]''` prints Gregory I "the Great" in the text. The index has a *see* reference from "Gregory the Great" to "Gregory I."

A few points for `\AKA` and `\AKA*` follow:

- With `\AKA`, `[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}` is the main name. The cross-reference is `[`⟨*Alt. FNN*⟩`]{`⟨*Alt. SNN*⟩`}[`⟨*Alt. names*⟩`]`. Please do not think that ⟨*Alt. FNN*⟩ belongs to the main name or `\AKA` will fail.
- `\AKA` fails with the old syntax: `\AKA{Louis}[XIV]{Sun King}`. `\AKA` and `\AKA*` fail with the old form: `\AKA{Gregory}[I]{Gregory}[the Great]`.
- The ⟨*Alt. SNN*⟩ field uses comma-delimited suffixes.
- The ⟨*Alt. names*⟩ field does not use comma-delimited suffixes.
- Eastern names work: One can refer to Lafcadio Hearn as KOIZUMI Yakumo: `\CapName\AKA[Lafcadio]{Hearn}{Koizumi, Yakumo}`.
- Particles work: Du Cange is the alternate name for Charles du Fresne, which is capitalized via `\CapThis\AKA`. See also Section 2.10.2.
- Reversing works, *e.g.* `\RevComma`: Hope, Leslie Townes.
- The name fields of `\PretagName` correspond with the `[`⟨*Alt. FNN*⟩`]{`⟨*Alt. SNN*⟩`}[`⟨*Alt. names*⟩`]` fields of `\AKA`: `\AKA{Vlad III, Dracula}{Vlad, Ţepeş}` corresponds with `\PretagName{Vlad, Ţepeş}{Vlad Tepes}`. It fails with `\PretagName{Vlad}[Ţepeş]{Vlad Tepes}`.

`\AKA` will not create multiple cross-references. Handle the special case where one moniker applies to multiple people with a manual solution, *e.g.*, "Snellius" for both Willebrord Snel van Royen and his son Rudolph Snel van Royen:

```
\index{Snellius|see{Snel van Royen, Rudolph; Snel van Royen, Willebrord}}
```

Cross-references generated by `\AKA` and `\AKA*` are meant only to be *see* references, never page entries. See also Section 2.2.4. In certain cases, the alternate name might need to be indexed with page numbers and *see also* references. Do not use `\AKA` in those cases, rather, consider the following:

- Refer to the person intended, *e.g.*, MAIMONIDES (Moses ben-Maimon): `\Name{Maimonides} (\AKA{Maimonides}{Moses ben-Maimon})`
- We now have a name and a *see* reference. Now one must refer to the alternate name, *e.g.*, RAMBAM: `\Name{Rambam}`.
- The alternate name must occur before making a cross-reference to the main name, in this case, Maimonides.
- Add `\index{Rambam|seealso{Maimonides}}` at the end of the document to ensure that it is the last entry among the cross-references. Generally, *see also* references follow *see* references in an index entry.[10]

Even with the new syntax, using makeindex may require some manual entries:

```
\index{Doctor Angelicus@\textit{Doctor Angelicus}%
|see{Thomas Aquinas}}%
\index{Thomas of Aquino|see{Thomas Aquinas}}%
Perhaps the greatest medieval theologian was \Name{Thomas, Aquinas}
(Thomas of Aquino), also known as \textit{Doctor Angelicus}. "Aquinas"
is not a surname.
```

Perhaps the greatest medieval theologian was THOMAS AQUINAS (Thomas of Aquino), also known as *Doctor Angelicus*. "Aquinas" is not a surname.

2.7.2 `\PName`

`\PName` `\PName` is a "convenience macro" meant for Western names. It generates a main name followed by a cross-reference in parentheses with the following syntax:

`\PName[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[`⟨*other FNN*⟩`]{`⟨*other SNN*⟩`}[`⟨*other alt.*⟩`]`

Although `\PName` creates an easy shortcut, its drawbacks are many. It only can use the ⟨*FNN*⟩⟨*SNN*⟩ form of `\AKA`. It cannot use `\AKA*`. `\PName` really is ill-suited to work with `\CapName`, `\CapThis`, `\RevComma`, `\RevName`, and the related package options. Use it as needed, and *caveat auctor*.

The author determines the name that is indexed (the first name) and the subsequent name that only occurs as a *see* reference. That subsequent name is never shortened in the text. To do that, using the table below, one would type, *e.g.*, `Arouet\IndexName{Voltaire}` or use the Rambam example above. `\PName` can generate the following examples:

`\PName[Mark]{Twain}[Samuel L.]{Clemens}`	MARK TWAIN (Samuel L. Clemens)
`\PName*[Mark]{Twain}[Samuel L.]{Clemens}`	Mark Twain (Samuel L. Clemens)
`\PName[Mark]{Twain}[Samuel L.]{Clemens}`	Twain (Samuel L. Clemens)
`\PName{Voltaire}[François-Marie]{Arouet}`	VOLTAIRE (François-Marie Arouet)
`\PName*{Voltaire}[François-Marie]{Arouet}`	Voltaire (François-Marie Arouet)
`\PName{Voltaire}[François-Marie]{Arouet}`	Voltaire (François-Marie Arouet)

If one used `\PName{William, I}[William]{the Conqueror}` the body text would look right but the index cross-reference would be in error. Medieval and Eastern names are not suited for `\PName`. For them use `\AKA`.

[10] Different standards exist for punctuating index entries and cross-references. Check with your publisher, style guide, docs for xindy and makeindex, and http://tex.stackexchange.com.

2.8 Indexing Macros

2.8.1 \IndexName

\IndexName This macro creates an index entry like those created by \Name and friends. It prints no text in the body and permits no special formatting. The syntax is:

\IndexName[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]

\IndexName complies with the new syntax. If ⟨*FNN*⟩ are absent, it indexes ⟨*Alternate names*⟩ as an affix using the old syntax; otherwise it ignores ⟨*Alternate names*⟩. If indexing is switched off (see Section 2.8.5), this macro does nothing. It will not create index entries for names used with \AKA as cross-references.

The indexing mechanism in the nameauth package follows *Chicago Manual of Style* standards regarding Western names and affixes. Thus the name Chesley B. Sullenberger III becomes "Sullenberger, Chesley B., III" in the index. This formatting only occurs for Western names (where ⟨*FNN*⟩ are present).

2.8.2 \TagName

\TagName This macro creates an index tag that will be appended to all index entries for a corresponding \Name from when it is invoked until the end of the document or a corresponding \UntagName. Both \TagName and \UntagName handle their arguments like \IndexName. If global tags are desired, tag names in the preamble.

\TagName[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]{⟨*tag*⟩}

Tags created by \TagName can be helpful in the indexes of history texts. Several features of this package are designed for historical research. Suppose you are working with medieval subject matter. The following macros come in handy:

`\TagName{Leo, I}{, pope}`
`\TagName{Gregory, I}{, pope}`
...
	(in the preamble)

`\Name*{Leo, I} \Name*{Gregory, I}` ...	(first references to Leo I and Gregory I)

`\Name*{Leo, I} was known as`
`\AKA{Leo, I}{Leo}[the Great].`
...
	Leo I was known as Leo the Great.

`\Name{Gregory, I} ``\AKA*{Gregory, I}%` `{Gregory}[the Great],'' another major` `pope.`	Gregory "the Great," another major pope.

Here \TagName causes the nameauth indexing macros to append "␣pope" to the index entries for Gregory I and Leo I.

Tags are literal text that can be daggers, asterisks, and even specials. For example, all fictional names in the index of this manual have an asterisk without any spaces before it. If space is desired between the entry and the tag, one must add the space at the start of the tag. Tagging aids scholarly indexing and can include life/regnal dates and other information.

\TagName works with all name types, not just medieval names. Back in Section 2.2 we had the example of Jimmy Carter (cross-reference in the index). \TagName adds "␣president" to his index entry.

You can use the {⟨*tag*⟩} field of \TagName to add specials to index entries for names. Every name in this document is tagged with at least {|hyperpage} to allow hyperlinks in the index using the ltxdoc class and hypdoc package.

2.8.3 \UntagName

\UntagName

\TagName will replace one tag with another tag, but it does not remove a tag from a name. That is the role of \UntagName. The syntax is:

\UntagName[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]

By using \TagName and \UntagName, one can disambiguate different people with the same name. For example:

```
This refers to \Name*[John]{Smith}.
Now another \ForgetName[John]{Smith}%
\TagName[John]{Smith}{ (other)}\Name[John]{Smith}.
Then a third \ForgetName[John]{Smith}%
\TagName[John]{Smith}{ (third)}\Name[John]{Smith}.
Then the first \UntagName[John]{Smith}\Name*[John]{Smith}.
```

This refers to John Smith.	*index*: Smith, John
Now another John Smith.	*index*: Smith, John (second)
Then a third John Smith.	*index*: Smith, John (third)
Then the first John Smith.	*index*: Smith, John

The tweaking macros \ForgetName and \SubvertName make it seem like you are dealing with three people who have the same name. The index tags will group together those entries with the same tag.[11]

Please remember to note the differences between \TagName and \UntagName on the one hand and \PretagName on the other. See Section 2.5.8.

2.8.4 Global Name Exclusion

\ExcludeName

This macro globally prevents the indexing of a particular name or cross-reference. If you do not use it at the beginning of the document, you may not exclude any name or cross-reference that has been used already. The syntax is:

\ExcludeName[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]

For example, \ExcludeName[Kris]{Kringle} will permit Kris Kringle and Kringle to appear in the body text via \Name[Kris]{Kringle}, but no index entry can occur for this name. \ExcludeName[Santa]{Claus} will prevent \AKA[Kris]{Kringle}[Santa]{Claus} Santa Claus from generating a cross-reference in the index. Instead of the global \ExcludeName, it is likelier that you would enclose \Name, etc. between \IndexInactive and \IndexActive.

[11] Since this document, unlike the example above, puts an asterisk by all fictional names in the index, it puts an asterisk at the beginning of the tags above and does not \UntagName John Smith, but retags him with an asterisk again.

2.8.5 Indexing Control

\IndexActive
\IndexInactive

Using the `noindex` option deactivates the indexing function of this package until `\IndexActive` occurs. Another macro, `\IndexInactive`, will deactivate indexing again. These can be used throughout the document, independently of `\ExcludeName`. They are global in scope, as are the other toggle macros in this package, so one must be explicit in turning indexing on and off.

2.9 Variant Spellings

This section illustrates why this package is called "nameauth." Here we get to an example where the macros work together to implement a name authority.

Handling variant name spellings can be complicated. For example, let us assume that you are editing a collection of essays. You might settle on the form W.E.B. Du Bois in your name authority. An essay in that collection might use the alternate spelling W.E.B. DuBois. The author or publisher who owns that work might not grant you permission to alter the spelling. In that case, you could add an alternate spelling. Using the simplified interface, it would be:

```
\begin{nameauth}
  \< DuBois & W.E.B. & Du Bois & >
  \< AltDuBois & W.E.B. & DuBois & >
\end{nameauth}
```

If you wanted to index the alternate spelling with its own entry, the trivial use of `\AltDuBois` allows that easily. All you need do is make cross-references to each variant in the index so that the reader is aware of them.

Nevertheless, Du Bois and DuBois differ only by spaces. For several good reasons, such as fault tolerance in typing, the first/subsequent use mechanism ignores spaces and sees them as *the same name*. Use `\ForgetName[W.E.B.]{Du Bois}` to trigger the first use of `\AltDuBois` in that section.

If you wanted to index the variants under only one name entry, it gets more complicated. You could do the following:

1. Use `\ForgetName[W.E.B.]{Du Bois}` at the start of the section.
2. Wrap `\AltDuBois` between `\IndexInactive` and `\IndexActive`.
3. Call `\IndexName` with the authoritative form right after `\IndexActive`.
4. Create a cross-reference in the index.

This can be automated at the start of the section with something like:

```
\ForgetName[W.E.B.]{DuBois}
\gdef\OtherDuBois{\IndexInactive\AltDuBois\IndexActive%
  \IndexName[W.E.B.]{Du Bois}}
\index{DuBois, W.E.B.|see{Du Bois, W.E.B.}}
```

The alternate section mentions `\OtherDuBois` starting with a first use: W.E.B. DuBois. Subsequent uses of `\OtherDuBois` print DuBois. Of course, one could get more complex than the example above. The index will only hold the standard entry for W.E.B. Du Bois: "Du Bois, W.E.B." and a cross-reference from the variant "DuBois, W.E.B." to the standard entry.

2.10 Naming Pattern Reference

2.10.1 Basic Naming

Western Names

First reference in the text:
John Smith
	`\Name*[John]{Smith}`
	`\Name[John]{Smith}`
	`\FName[John]{Smith}`

Subsequent full: John Smith	`\Name*[John]{Smith}`
Subsequent surname: Smith	`\Name[John]{Smith}`
Subsequent forename: John	`\FName[John]{Smith}`

Long first reference:
Jane Q. Public
	`\Name*[J.Q.]{Public}[Jane Q.]`
	`\Name[J.Q.]{Public}[Jane Q.]`
	`\FName[J.Q.]{Public}[Jane Q.]`

Subsequent full: J.Q. Public	`\Name*[J.Q.]{Public}`
Alternate: Jane Qetsiyah Public	`\Name*[J.Q.]{Public}[Jane Qetsiyah]`
Alternate: Janie	`\FName[J.Q.]{Public}[Janie]`

Western Plus Affixes

Always use a comma to delimit name/affix pairs.

First reference:
George S. Patton Jr.
	`\Name*[George S.]{Patton, Jr.}`
	`\Name[George S.]{Patton, Jr.}`
	`\FName[George S.]{Patton, Jr.}`

Subsequent: George S. Patton Jr.	`\Name*[George S.]{Patton, Jr.}`
Subsequent surname: Patton	`\Name[George S.]{Patton, Jr.}`
Subsequent forename: George	`\FName[George S.]{Patton, Jr.}[George]`

```
\begin{nameauth}
  \< Smith & John & Smith & >
  \< JQP & J.Q. & Public & >
  \< Patton & George S. & Patton, Jr. & >
\end{nameauth}
```

`\Smith, \LSmith, \Smith, and \SSmith`:
John Smith, John Smith, Smith, and John

`\JQP[Jane Q.], \LJQP[Jane Q.], and \JQP[Jane Q.]`:
Jane Q. Public, Jane Q. Public, and Public

`\LJQP[Jane Qetsiyah]\ and \SJQP[Janie]`:
Jane Qetsiyah Public and Janie

`\Patton, \LPatton, \Patton, and \SPatton`:
George S. Patton Jr., George S. Patton Jr., Patton, and George S.

`\SPatton[George]` prints George.

New Syntax: Royal, Eastern, and Ancient

Using `\Name{Demetrius, I Soter}` keeps the number with the affix. To keep the number with the name, use `\Name{Demetrius I, Soter}`. See also Section 2.4.1.

First reference: Francis I	`\Name*{Francis, I}`
	`\Name{Francis, I}`
	`\FName{Francis, I}`

Subsequent full: Francis I	`\Name*{Francis, I}`
Subsequent name: Francis	`\Name{Francis, I}` `\FName{Francis, I}`

First reference: Demetrius I Soter	`\Name*{Demetrius, I Soter}`
	`\Name{Demetrius, I Soter}`
	`\FName{Demetrius, I Soter}`

Subsequent full: Demetrius I Soter	`\Name*{Demetrius, I Soter}`
Subsequent name: Demetrius	`\Name{Demetrius, I Soter}` `\FName{Demetrius, I Soter}`

First reference: Sun Yat-sen	`\Name*{Sun, Yat-sen}`
	`\Name{Sun, Yat-sen}`
	`\FName{Sun, Yat-sen}`

Subsequent full: Sun Yat-sen	`\Name*{Sun, Yat-sen}`
Subsequent name: Sun	`\Name{Sun, Yat-sen}` `\FName{Sun, Yat-sen}`

First mononym reference: Plato	`\Name*{Plato}`
	`\Name{Plato}`
	`\FName{Plato}`

Subsequent: Plato	`\Name*{Plato}`
	`\Name{Plato}`
	`\FName{Plato}`

```
\begin{nameauth}
  \< Francis & & Francis, I & >
  \< Dem & & Demetrius, I Soter & >
  \< Sun & & Sun, Yat-sen & >
  \< Plato & & Plato & >
\end{nameauth}
```

`\Francis, \LFrancis, \Francis, and \SFrancis`:
Francis I, Francis I, Francis, and Francis

`\Dem, \LDem, \Dem, and \SDem`:
Demetrius I Soter, Demetrius I Soter, Demetrius, and Demetrius

`\Sun, \LSun, \Sun, and \SSun`:
Sun Yat-sen, Sun Yat-sen, Sun, and Sun

`\Plato, \LPlato, \Plato, and \SPlato`:
Plato, Plato, Plato, and Plato.

You also can "stack" `\CapThis`, `\CapName`, `\RevName`, `\KeepAffix`, and so on in front of these control sequences. `\CapName\LSun` generates SUN Yat-sen.

Old Syntax: Royal and Eastern

Avoid these forms except with the `comma` option. `\Name{Ptolemy}[I Soter]` keeps the number with the affix. Use `\Name{Ptolemy I}[Soter]` to keep the number with the name. See also Section 2.4.1.

First reference: Henry VIII	`\Name*{Henry}[VIII]`
	`\Name{Henry}[VIII]`
	`\FName{Henry}[VIII]`

Subsequent full: Henry VIII	`\Name*{Henry}[VIII]`
Subsequent name: Henry	`\Name{Henry}[VIII]` `\FName{Henry}[VIII]`

First reference: Ptolemy I Soter	`\Name*{Ptolemy}[I Soter]`
	`\Name{Ptolemy}[I Soter]`
	`\FName{Ptolemy}[I Soter]`

Subsequent full: Ptolemy I Soter	`\Name*{Ptolemy}[I Soter]`
Subsequent name: Ptolemy	`\Name{Ptolemy}[I Soter]` `\FName{Ptolemy}[I Soter]`
First reference: Mao Tse-tung	`\Name*{Mao}[Tse-tung]` `\Name{Mao}[Tse-tung]`
Subsequent full: Mao Tse-tung	`\Name*{Mao}[Tse-tung]`
Subsequent name: Mao	`\Name{Mao}[Tse-tung]` `\FName{Mao}[Tse-tung]`

```
\begin{nameauth}
  \< Henry & & Henry & VIII >
  \< Ptol & & Ptolemy & I Soter >
  \< Mao & & Mao & Tse-tung >
\end{nameauth}
```

`\Henry, \LHenry, \Henry, and \SHenry`:
Henry VIII, Henry VIII, Henry, and Henry

`\Ptol, \LPtol, \Ptol, and \SPtol`:
Ptolemy I Soter, Ptolemy I Soter, Ptolemy, and Ptolemy

`\Mao, \LMao, \Mao, and \SMao`:
Mao Tse-tung, Mao Tse-tung, Mao, and Mao

Avoid mixing old and new syntax. In the body text, `\Name{Antiochus, IV}` and `\Name{Antiochus, IV}[Epiphanes]` look alike, but their index entries differ.

- Use `\Name{Antiochus, IV Epiphanes}` to get Antiochus IV Epiphanes and Antiochus in the text and "Antiochus IV Epiphanes" in the index.
- Use `\Name{Antiochus~IV, Epiphanes}` to get Antiochus IV Epiphanes and Antiochus IV in the text and "Antiochus IV Epiphanes" in the index.
- Use `\Name{Antiochus, IV}` to get Antiochus IV and Antiochus in the text. Use something like `\TagName{Antiochus, IV}{ Epiphanes}` to get "Antiochus IV Epiphanes" in the index and add "Epiphanes" in the text.

2.10.2 Particles

The following illustrate the American style of particulate names.

First: Walter de la Mare	`\Name*[Walter]{de la Mare}`
	`\Name[Walter]{de la Mare}`
	`\FName[Walter]{de la Mare}`

Subsequent: de la Mare	`\Name[Walter]{de la Mare}`
Start of sentence: De la Mare	`\CapThis\Name[Walter]{de la Mare}`
Forename: Walter	`\FName[Walter]{de la Mare}`

The Continental style differs slightly. These first three forms below put the particles in the index. Long macros are split for readability.

The (admittedly long) first use: Johann Wolfgang von Goethe	`\Name*[Johann Wolfgang von]{Goethe}`
	`\Name[Johann Wolfgang von]{Goethe}`
	`\FName[Johann Wolfgang von]{Goethe}`

Subsequent: Goethe	`\Name[Johann Wolfgang von]{Goethe}`
Forenames: Johann Wolfgang	`\FName[Johann Wolfgang von]{Goethe}%` `[Johann Wolfgang]`

These latter examples of the Continental style use the nickname feature to omit the particles from the index.

First: Adolf von Harnack	`\Name*[Adolf]{Harnack}[Adolf von]`
	`\Name[Adolf]{Harnack}[Adolf von]`
	`\FName[Adolf]{Harnack}[Adolf von]`

Subsequent full: Adolf von Harnack	`\Name*[Adolf]{Harnack}[Adolf von]`
Subsequent surname: Harnack	`\Name[Adolf]{Harnack}[Adolf von]` `\Name[Adolf]{Harnack}`
Subsequent forename: Adolf	`\FName[Adolf]{Harnack}`

```
\begin{nameauth}
  \< DLM & Walter & de la Mare & >
  \< JWG & Johann Wolfgang von & Goethe & >
  \< Harnack & Adolf & Harnack & >
\end{nameauth}
```

`\DLM\ and \CapThis\DLM`:
Walter de la Mare and De la Mare.

`\JWG\ and \JWG`:
Johann Wolfgang von Goethe and Goethe.

`\Harnack[Adolf von]\ and \Harnack`:
Adolf von Harnack and Harnack

You will not see Harnack's "von" in the index because it was used only in the alternate forenames field.

3 Implementation

3.1 Boolean Values

We begin with Boolean values that control printed name forms. `@nameauth@FullName` toggles long or short forms in subsequent name uses. `@nameauth@FirstName` is used when printing only first names. `@nameauth@AltAKA` determines if `\AKA` will print a longer or shorter name, depending on the starred or unstarred form.

```
\newif\if@nameauth@FullName
\newif\if@nameauth@FirstName
\newif\if@nameauth@AltAKA
```

The next few Boolean values control formatting. The first is toggled with `\NamesActive` and `\NamesInctive` or the front and main matter options. The next toggles the formatting of first occurrences of names. The last forces name formatting whenever formatting is active, or allows normal operation.

```
\newif\if@nameauth@DoFormat
\newif\if@nameauth@FirstFormat
\newif\if@nameauth@AlwaysFormat
```

The `comma` and `nocomma` options toggle the first value below, while `\ShowComma` toggles the second. Each instance of `\Name` and `\AKA` reset `@nameauth@ShowComma`.

```
\newif\if@nameauth@AlwaysComma
\newif\if@nameauth@ShowComma
```

`\KeepAffix` toggles the value below. Each instance of `\Name` and `\AKA` reset it.

```
\newif\if@nameauth@NBSP
```

`\IndexActive` and `\IndexInctive` toggle the value below.

```
\newif\if@nameauth@DoIndex
```

The `pretag` and `nopretag` options toggle the value below.

```
\newif\if@nameauth@Pretag
```

This Boolean value is used for detection of affixes and final periods.

```
\newif\if@nameauth@Punct
```

This Boolean value is triggered by `\CapThis`.

```
\newif\if@nameauth@DoCaps
```

The next Boolean values govern full name capitalization, name reversing, and name reversing with commas.

```
\newif\if@nameauth@AllCaps
\newif\if@nameauth@AllThis
\newif\if@nameauth@RevAll
\newif\if@nameauth@RevThis
\newif\if@nameauth@RevAllComma
\newif\if@nameauth@RevThisComma
```

3.2 Package Options

The following package options interact with many of the prior Boolean values. Suppressing and showing commas is set at load time. Most options can be changed with user interface macros. Avoid changing the internal Boolean values directly.

```
\newcommand{\NamesFormat}{}
\def\@nameauth@Actual{@}
\DeclareOption{mainmatter}{\@nameauth@DoFormattrue}
\DeclareOption{frontmatter}{\@nameauth@DoFormatfalse}
\DeclareOption{smallcaps}{\renewcommand{\NamesFormat}{\scshape}}
\DeclareOption{italic}{\renewcommand{\NamesFormat}{\itshape}}
\DeclareOption{boldface}{\renewcommand{\NamesFormat}{\bfseries}}
\DeclareOption{noformat}{\renewcommand{\NamesFormat}{}}
\DeclareOption{alwaysformat}{\@nameauth@AlwaysFormattrue}
\DeclareOption{allcaps}{\@nameauth@AllCapstrue}
\DeclareOption{normalcaps}{\@nameauth@AllCapsfalse}
\DeclareOption{allreversed}%
  {\@nameauth@RevAlltrue\@nameauth@RevAllCommafalse}
\DeclareOption{allrevcomma}%
  {\@nameauth@RevAlltrue\@nameauth@RevAllCommatrue}
\DeclareOption{notreversed}%
  {\@nameauth@RevAllfalse\@nameauth@RevAllCommafalse}
\DeclareOption{comma}{\@nameauth@AlwaysCommatrue}
\DeclareOption{nocomma}{\@nameauth@AlwaysCommafalse}
\DeclareOption{index}{\@nameauth@DoIndextrue}
\DeclareOption{noindex}{\@nameauth@DoIndexfalse}
\DeclareOption{pretag}{\@nameauth@Pretagtrue}
\DeclareOption{nopretag}{\@nameauth@Pretagfalse}
\ExecuteOptions%
  {nocomma,%
   mainmatter,%
   index,%
   pretag,%
   normalcaps,%
   notreversed,%
   smallcaps}
\ProcessOptions\relax
```

Now we load the required packages. They facilitate the first/subsequent name uses, the parsing of arguments, and the implementation of starred forms.

```
\RequirePackage{etoolbox}
\RequirePackage{trimspaces}
\RequirePackage{suffix}
\RequirePackage{xargs}
```

3.3 Internal Macros

`\@nameauth@Clean` Thanks to Heiko Oberdiek, this macro produces a "sanitized" string, even using accented characters, based on the parameters of `\Name` and friends. With this we can construct a control sequence name and test for it to determine the existence of pseudonyms and the first or subsequent occurrences of a name.

```
\newcommand*{\@nameauth@Clean}[1]%
  {\expandafter\zap@space\detokenize{#1} \@empty}
```

`\@nameauth@Root` The following two macros parse ⟨*SNN*⟩ into a radix and a comma-delimited suffix, returning only the radix. They (and their parameters) are expandable in order to facilitate proper indexing functionality. They form the kernel of the suffix removal and comma suppression features.

```
\newcommand*{\@nameauth@Root}[1]%
  {\@nameauth@TrimRoot#1,\@empty\relax}
```

`\@nameauth@TrimRoot` Throw out the comma and suffix, return the radix.

```
\def\@nameauth@TrimRoot#1,#2\relax{\trim@spaces{#1}}
```

`\@nameauth@CapRoot` The next two macros implement the particulate name capitalization mechanism by returning a radix where the first letter is capitalized.

```
\newcommand*{\@nameauth@CapRoot}[1]%
  {\@nameauth@CR#1\relax}
```

`\@nameauth@CR` Grab the first letter as one parameter, and everything before `\relax` as the second. Capitalize the first and return it with the second.

```
\def\@nameauth@CR#1#2\relax{\uppercase{#1}\@nameauth@Root{#2}}
```

`\@nameauth@AllCapRoot` This macro returns a fully-capitalized radix. It is used for generating capitalized Eastern family names in the body text.

```
\newcommand*{\@nameauth@AllCapRoot}[1]%
  {\uppercase{\@nameauth@Root{#1}}}
```

`\@nameauth@Suffix` The following two macros parse ⟨*SNN*⟩ into a radix and a comma-delimited suffix, returning only the suffix. Anything before a comma is stripped off by `\@nameauth@Suffix`, but a comma must be present in the parameter. Leading spaces are removed to allow consistent formatting.

```
\newcommand*{\@nameauth@Suffix}[1]%
  {\@nameauth@TrimSuffix#1\relax}
```

`\@nameauth@TrimSuffix` Throw out the radix, comma, and `\relax`; return the suffix with no leading spaces.

```
\def\@nameauth@TrimSuffix#1,#2\relax{\trim@spaces{#2}}
```

`\@nameauth@TestDot` This macro, based on a snippet by Uwe Lueck, checks for a period at the end of its parameter. It determines whether we need to call `\@nameauth@CheckDot` below.

```
\newcommand*{\@nameauth@TestDot}[1]%
{%
  \def\TestDot##1.\TestEnd##2\TestStop{\TestPunct{##2}}%
  \def\TestPunct##1%
    {\ifx\TestPunct##1\TestPunct\else\@nameauth@Puncttrue\fi}%
  \@nameauth@Punctfalse%
  \TestDot#1\TestEnd.\TestEnd\TestStop%
}
```

`\@nameauth@CheckDot` We assume that `\expandafter` precedes the invocation of `\@nameauth@CheckDot`, which only is called if the terminal character of the input is a period. We evaluate the lookahead `\@token` while keeping it on the list of input tokens.

```
\newcommand*{\@nameauth@CheckDot}%
  {\futurelet\@token\@nameauth@EvalDot}
```

`\@nameauth@EvalDot` If `\@token` is a full stop, we gobble the token.

```
\newcommand*{\@nameauth@EvalDot}%
  {\let\@period=.\ifx\@token\@period\expandafter\@gobble \fi}
```

`\@nameauth@FmtName` The following macros format the output of `\Name`, etc. `\@nameauth@FmtName` prints names in the body text, either formatted or not. Notice how `\NamesFormat` (Section 2.5.9) sits between a `\bgroup` and an `\egroup` to localize the font change. `@nameauth@AlwaysFormat` will force formatting when possible.

```
\newcommand*{\@nameauth@FmtName}[1]%
{%
  \if@nameauth@AlwaysFormat\@nameauth@FirstFormattrue\fi
  \@nameauth@TestDot{#1}%
  \if@nameauth@DoFormat
    \if@nameauth@FirstFormat
      \bgroup\NamesFormat{#1}\egroup%
    \else
      #1%
    \fi
  \else
    #1%
  \fi
}
```

`\@nameauth@Index` If the indexing flag is true, create an index entry, otherwise do nothing.

```
\newcommand*{\@nameauth@Index}[2]%
{%
  \def\cseq{#1}%
  \ifcsname\cseq!TAG\endcsname
    \ifcsname\cseq!PRE\endcsname
      \if@nameauth@DoIndex
        \index{\csname\cseq!PRE\endcsname#2\csname\cseq!TAG\endcsname}%
      \fi
    \else
      \if@nameauth@DoIndex\index{#2\csname\cseq!TAG\endcsname}\fi
    \fi
  \else
    \ifcsname\cseq!PRE\endcsname
      \if@nameauth@DoIndex\index{\csname\cseq!PRE\endcsname#2}\fi
    \else
      \if@nameauth@DoIndex\index{#2}\fi
    \fi
  \fi
}
```

3.4 User Interface Macros

The following macros have been documented previously. Either they call the internal macros or they set Boolean values.

`\CapThis` Capitalize first letter of name.

```
\newcommand*{\CapThis}{\@nameauth@DoCapstrue}
```

`\CapName` Capitalize entire name.

```
\newcommand*{\CapName}{\@nameauth@AllThistrue}
```

`\RevName` Reverse name order.

```
\newcommand*{\RevName}{\@nameauth@RevThistrue}
```

`\RevComma` Last name, comma, first name.

```
\newcommand*{\RevComma}%
  {\@nameauth@RevThistrue\@nameauth@RevThisCommatrue}
```

`\ShowComma` Put comma between name and suffix one time.

```
\newcommand*{\ShowComma}{\@nameauth@ShowCommatrue}
```

`\KeepAffix` Trigger a name-suffix pair to be separated by a non-breaking space.

```
\newcommand*{\KeepAffix}{\@nameauth@NBSPtrue}
```

`\NamesInactive` Switch to the "non-formatted" species of names.

```
\newcommand*{\NamesInactive}{\@nameauth@DoFormatfalse}
```

`\NamesActive` Switch to the "formatted" species of names.

```
\newcommand*{\NamesActive}{\@nameauth@DoFormattrue}
```

`\AllCapsInactive` Turn off global surname capitalization.

```
\newcommand*{\AllCapsInactive}{\@nameauth@AllCapsfalse}
```

`\AllCapsActive` Turn on global surname capitalization.

```
\newcommand*{\AllCapsActive}{\@nameauth@AllCapstrue}
```

`\ReverseInactive` Turn off global name reversing.

```
\newcommand*{\ReverseInactive}{\@nameauth@RevAllfalse}
```

`\ReverseActive` Turn on global name reversing.

```
\newcommand*{\ReverseActive}{\@nameauth@RevAlltrue}
```

`\ReverseCommaInactive` Turn off global "last-name-comma-first."

```
\newcommand*{\ReverseCommaInactive}%
  {\@nameauth@RevAllfalse\@nameauth@RevAllCommafalse}
```

`\ReverseCommaActive` Turn on global "last-name-comma-first."

```
\newcommand*{\ReverseCommaActive}%
  {\@nameauth@RevAlltrue\@nameauth@RevAllCommatrue}
```

`\IndexInactive` turn off global indexing of names.

```
\newcommand*{\IndexInactive}{\@nameauth@DoIndexfalse}
```

`\IndexActive` turn on global indexing of names.

```
\newcommand*{\IndexActive}{\@nameauth@DoIndextrue}
```

`\IndexActual` turn on global indexing of names.

```
\newcommand*{\IndexActual}[1]{\gdef\@nameauth@Actual{#1}}
```

`\Name` Here is the heart of the package. Marc van Dongen provided the basic structure. Parsing, indexing, and formatting are in discrete elements.

```
\newcommandx*\Name[3][1=\@empty, 3=\@empty]%
{%
  \let\ex\expandafter%
  \leavevmode\hbox{}%
```

Names occur in horizontal mode; we ensure that.

```
  \protected@edef\testa{#1}%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\testb{\trim@spaces{#2}}%
  \protected@edef\testbr{\@nameauth@Root{#2}}%
  \protected@edef\testc{#3}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \Name: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \AKA: Essential name malformed}%
    \fi
  \fi
```

We make copies of the arguments to test them and make parsing decisions.

```
  \if@nameauth@AllCaps\@nameauth@AllThistrue\fi
  \if@nameauth@RevAll\@nameauth@RevThistrue\fi
  \if@nameauth@RevAllComma\@nameauth@RevThisCommatrue\fi
```

If global caps. reversing, and commas are true, set the local flags true.

The code below handles non-breaking and regular spaces, as well as commas, in the text and the index by setting up which kind we want to use. These will be inserted as appropriate as the output is formatted.

```
  \protected@edef\ISpace{\space}%
  \protected@edef\Space{\space}%
  \if@nameauth@NBSP\protected@edef\Space{\nobreakspace}\fi
  \if@nameauth@AlwaysComma
    \protected@edef\ISpace{,\space}%
    \protected@edef\Space{,\space}%
    \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
  \fi
  \if@nameauth@ShowComma
    \protected@edef\ISpace{,\space}%
    \protected@edef\Space{,\space}%
    \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
  \fi
```

The section below parses any “surnames” into name/suffix pairs and figures out how to capitalize and reverse them as needed, storing the results for the main parser.

```
\protected@edef\RawShort{\@nameauth@Root{#2}}%
\protected@edef\CapShort{\@nameauth@CapRoot{#2}}%
\protected@edef\AllCapShort{\@nameauth@AllCapRoot{#2}}%
\let\IndexShort\RawShort%
\ifx\testb\testbr
  \protected@edef\Suff{\@empty}%
  \let\IndexSNN\RawShort%
  \let\Reversed\RawShort%
  \let\SNN\RawShort%
  \let\PrintShort\RawShort%
  \if@nameauth@DoCaps
    \let\Reversed\CapShort%
    \let\SNN\CapShort%
    \let\PrintShort\CapShort%
  \fi
  \if@nameauth@AllThis
    \let\Reversed\AllCapShort%
    \let\SNN\AllCapShort%
    \let\PrintShort\AllCapShort%
  \fi
\else
  \protected@edef\Suff{\@nameauth@Suffix{#2}}%
  \protected@edef\IndexSNN{\RawShort\ISpace\Suff}%
  \protected@edef\Reversed{\Suff\Space\RawShort}%
  \protected@edef\SNN{\RawShort\Space\Suff}%
  \if@nameauth@RevThis
    \let\PrintShort\Suff%
  \else
    \let\PrintShort\RawShort%
  \fi
  \if@nameauth@DoCaps
    \protected@edef\Reversed{\Suff\Space\CapShort}%
    \protected@edef\SNN{\CapShort\Space\Suff}%
    \if@nameauth@RevThis
      \let\PrintShort\Suff%
    \else
      \let\PrintShort\CapShort%
    \fi
  \fi
  \if@nameauth@AllThis
    \protected@edef\Reversed{\Suff\Space\AllCapShort}%
    \protected@edef\SNN{\AllCapShort\Space\Suff}%
    \if@nameauth@RevThis
      \let\PrintShort\Suff%
    \else
      \let\PrintShort\AllCapShort%
    \fi
  \fi
\fi
```

Here we parse names.

```
\ifx\testa\@empty
  \ifx\testc\@empty
```

This is the section for momonyms, royal name/suffix pairs, and native Eastern names where comma-delimited suffixes are used. The first conditional below checks if we are trying to use an alternate name cross-reference as a main name (code !PN for pseudonym). If we are using a legitimate name, we generate an index entry.

```
    \ifcsname\csb!PN\endcsname
      \PackageWarning{nameauth}%
      {macro \Name: Xref: #2 cannot be a page reference}%
    \else
      \@nameauth@Index{\csb}{\IndexSNN}%
    \fi
```

If formatting is active, we handle first and subsequent formatting of names in the main matter (code !MN for main matter name). First we handle subsequent uses. We need `\expandafter` to enable the punctuation detection.

```
    \if@nameauth@DoFormat
      \ifcsname\csb!MN\endcsname
        \if@nameauth@FirstName
          \@nameauth@FullNamefalse%
          \@nameauth@FirstNamefalse%
        \fi
        \if@nameauth@FullName
          \@nameauth@FullNamefalse%
          \if@nameauth@RevThis
            \ex\@nameauth@FmtName\ex{\Reversed}%
          \else
            \ex\@nameauth@FmtName\ex{\SNN}%
          \fi
        \else
          \ex\@nameauth@FmtName\ex{\PrintShort}%
        \fi
      \else
```

Handle first uses.

```
        \@nameauth@FirstFormattrue%
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNamefalse%
        \csgdef{\csb!MN}{}%
        \if@nameauth@RevThis
          \ex\@nameauth@FmtName\ex{\Reversed}%
        \else
          \ex\@nameauth@FmtName\ex{\SNN}%
        \fi
      \fi
    \else
```

Take care of names in the front matter (code !NF for non-formatted). First handle subsequent uses.

```
      \ifcsname\csb!NF\endcsname
        \if@nameauth@FirstName
          \@nameauth@FullNamefalse%
          \@nameauth@FirstNamefalse%
        \fi
        \if@nameauth@FullName
          \@nameauth@FullNamefalse%
          \if@nameauth@RevThis
            \ex\@nameauth@FmtName\ex{\Reversed}%
          \else
            \ex\@nameauth@FmtName\ex{\SNN}%
          \fi
        \else
          \ex\@nameauth@FmtName\ex{\PrintShort}%
        \fi
      \else
```

Handle first uses.

```
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNamefalse%
        \csgdef{\csb!NF}{}%
        \if@nameauth@RevThis
          \ex\@nameauth@FmtName\ex{\Reversed}%
        \else
          \ex\@nameauth@FmtName\ex{\SNN}%
        \fi
      \fi
    \fi
  \else
```

This is the section that handles the old syntax for royal names and native Eastern names. The first conditional below checks if we are trying to use an alternate name cross-reference as a main name (code !PN for pseudonym). If we are using a legitimate name, we generate an index entry.

```
    \ifcsname\csbc!PN\endcsname
      \PackageWarning{nameauth}%
      {macro \Name: Xref: #2 #3 cannot be a page reference}%
    \else
      \@nameauth@Index{\csbc}{\IndexSNN\ISpace\argc}%
    \fi
```

If formatting is active, we handle first and subsequent formatting of names in the main matter (code !MN for main matter name). First we handle subsequent uses.

```
    \if@nameauth@DoFormat
      \ifcsname\csbc!MN\endcsname
        \if@nameauth@FirstName
          \@nameauth@FullNamefalse%
          \@nameauth@FirstNamefalse%
        \fi
        \if@nameauth@FullName
          \@nameauth@FullNamefalse%
          \if@nameauth@RevThis
            \ex\@nameauth@FmtName\ex{\ex\argc\ex\space\SNN}%
          \else
            \ex\@nameauth@FmtName\ex{\ex\SNN\ex\space\argc}%
          \fi
        \else
          \if@nameauth@RevThis
            \ex\@nameauth@FmtName\ex{\argc}%
          \else
            \ex\@nameauth@FmtName\ex{\PrintShort}%
          \fi
        \fi
      \else
```

Handle first uses.

```
        \@nameauth@FirstFormattrue%
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNamefalse%
        \csgdef{\csbc!MN}{}%
        \if@nameauth@RevThis
          \ex\@nameauth@FmtName\ex{\ex\argc\ex\space\SNN}%
        \else
          \ex\@nameauth@FmtName\ex{\ex\SNN\ex\space\argc}%
        \fi
      \fi
    \else
```

Take care of names in the front matter (code !NF for non-formatted). First handle subsequent uses.

```
      \ifcsname\csbc!NF\endcsname
        \if@nameauth@FirstName
          \@nameauth@FullNamefalse%
          \@nameauth@FirstNamefalse%
        \fi
        \if@nameauth@FullName
          \@nameauth@FullNamefalse%
          \if@nameauth@RevThis
            \ex\@nameauth@FmtName\ex{\ex\argc\ex\space\SNN}%
          \else
            \ex\@nameauth@FmtName\ex{\ex\SNN\ex\space\argc}%
          \fi
        \else
          \if@nameauth@RevThis
            \ex\@nameauth@FmtName\ex{\argc}%
          \else
            \ex\@nameauth@FmtName\ex{\PrintShort}%
          \fi
        \fi
      \else
```

Handle first uses.

```
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNamefalse%
        \csgdef{\csbc!NF}{}%
        \if@nameauth@RevThis
          \ex\@nameauth@FmtName\ex{\ex\argc\ex\space\SNN}%
        \else
          \ex\@nameauth@FmtName\ex{\ex\SNN\ex\space\argc}%
        \fi
      \fi
    \fi
  \fi
\else
```

This is the section that handles Western names and non-native Eastern names. The first pair of conditionals handle the `comma` option, `\evThisComma`, and alternate forenames. The next conditional below checks if we are trying to use an alternate name cross-reference as a main name (code !PN for pseudonym). If we are using a legitimate name, we generate an index entry.

```
\if@nameauth@RevThisComma
  \protected@edef\ISpace{,\space}%
  \protected@edef\Space{,\space}%
  \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
\fi
\ifx\testc\@empty
  \let\FNN\arga%
\else
  \let\FNN\argc%
\fi
\ifcsname\csab!PN\endcsname
  \PackageWarning{nameauth}%
  {macro \Name: Xref: #1 #2 cannot be a page reference}%
\else
  \ifx\Suff\@empty
    \@nameauth@Index{\csab}{\IndexShort,\space\arga}%
  \else
    \@nameauth@Index{\csab}{\IndexShort,\space\arga,\space\Suff}%
  \fi
\fi
```

If formatting is active, we handle first and subsequent formatting of names in the main matter (code !MN for main matter name). First we handle subsequent uses.

```
\if@nameauth@DoFormat
  \ifcsname\csab!MN\endcsname
    \if@nameauth@FirstName
      \@nameauth@FullNamefalse%
      \@nameauth@FirstNamefalse%
      \let\PrintShort\FNN%
    \fi
    \if@nameauth@FullName
      \@nameauth@FullNamefalse%
      \if@nameauth@RevThis
        \ex\@nameauth@FmtName\ex{\ex\SNN\ex\Space\FNN}%
      \else
        \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNN}%
      \fi
    \else
      \ex\@nameauth@FmtName\ex{\PrintShort}%
    \fi
  \else
```

Handle first uses.

```
        \@nameauth@FirstFormattrue%
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNamefalse%
        \csgdef{\csab!MN}{}%
        \if@nameauth@RevThis
          \ex\@nameauth@FmtName\ex{\ex\SNN\ex\Space\FNN}%
        \else
          \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNN}%
        \fi
      \fi
    \else
```

Take care of names in the front matter (code !NF for non-formatted). First handle subsequent uses.

```
      \ifcsname\csab!NF\endcsname
        \if@nameauth@FirstName
          \@nameauth@FullNamefalse%
          \@nameauth@FirstNamefalse%
          \let\PrintShort\FNN%
        \fi
        \if@nameauth@FullName
          \@nameauth@FullNamefalse%
          \if@nameauth@RevThis
            \ex\@nameauth@FmtName\ex{\ex\SNN\ex\Space\FNN}%
          \else
            \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNN}%
          \fi
        \else
          \ex\@nameauth@FmtName\ex{\PrintShort}%
        \fi
      \else
```

Handle first uses.

```
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNamefalse%
        \csgdef{\csab!NF}{}%
        \if@nameauth@RevThis
          \ex\@nameauth@FmtName\ex{\ex\SNN\ex\Space\FNN}%
        \else
          \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNN}%
        \fi
      \fi
    \fi
  \fi
```

Reset all the "per name" Boolean values.

```
  \@nameauth@FirstFormatfalse%
  \@nameauth@NBSPfalse%
  \@nameauth@DoCapsfalse%
  \@nameauth@AllThisfalse%
  \@nameauth@ShowCommafalse%
  \@nameauth@RevThisfalse%
  \@nameauth@RevThisCommafalse%
```

Call the full stop detection.

```
  \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
}
```

`\Name*` `\Name*` sets a Boolean value and calls `\Name`.

```
\WithSuffix\def\Name*{\@nameauth@FullNametrue\Name}
```

`\FName` `\FName` sets a Boolean value and calls `\Name`.

```
\def\FName{\@nameauth@FirstNametrue\Name}
```

`\FName*` `\FName` and `\FName*` are identical.

```
\WithSuffix\def\FName*{\@nameauth@FirstNametrue\Name}
```

`\AKA` `\AKA` prints an alternate name and creates index cross-references. It prevents multiple generation of cross-references and suppresses double periods.

```
\newcommandx*\AKA[5][1=\@empty, 3=\@empty, 5=\@empty]%
{%
  \let\ex\expandafter%
  \leavevmode\hbox{}%
```

Names occur in horizontal mode; we ensure that.

```
  \protected@edef\testa{#1}%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\testb{\trim@spaces{#2}}%
  \protected@edef\testbr{\@nameauth@Root{#2}}%
  \protected@edef\testc{#3}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \protected@edef\testd{\trim@spaces{#4}}%
  \protected@edef\testdr{\@nameauth@Root{#4}}%
  \protected@edef\teste{#5}%
  \protected@edef\arge{\trim@spaces{#5}}%
  \def\csd{\@nameauth@Clean{#4}}%
  \def\csde{\@nameauth@Clean{#4#5}}%
  \def\cscd{\@nameauth@Clean{#3!#4}}%
```

Above we make copies of the arguments to test them and make parsing decisions.

```
\ifx\testb\@empty
  \PackageError{nameauth}%
  {macro \AKA: Essential name missing}%
\else
  \ifx\csb\@empty
    \PackageError{nameauth}%
    {macro \AKA: Essential name malformed}%
  \fi
\fi
\ifx\testd\@empty
  \PackageError{nameauth}%
  {macro \AKA: Essential name missing}%
\else
  \ifx\csd\@empty
    \PackageError{nameauth}%
    {macro \AKA: Essential name malformed}%
  \fi
\fi
```

We test for one kind of malformed input input.

```
\if@nameauth@AllCaps\@nameauth@AllThistrue\fi
\if@nameauth@RevAll\@nameauth@RevThistrue\fi
\if@nameauth@RevAllComma\@nameauth@RevThisCommatrue\fi
```

If global caps. reversing, and commas are true, set the local flags true.

The code below handles non-breaking and regular spaces, as well as commas, in the text and the index by setting up which kind we want to use. These will be inserted as appropriate as the output is formatted.

```
\protected@edef\ISpace{\space}%
\protected@edef\Space{\space}%
\if@nameauth@NBSP\protected@edef\Space{\nobreakspace}\fi
\if@nameauth@AlwaysComma
  \protected@edef\ISpace{,\space}%
  \protected@edef\Space{,\space}%
  \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
\fi
\if@nameauth@ShowComma
  \protected@edef\ISpace{,\space}%
  \protected@edef\Space{,\space}%
  \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
\fi
```

The section below parses any “surnames” into name/suffix pairs and figures out how to capitalize and reverse them as needed, storing the results for the main parser. We have to handle several more combinations here than with `\Name` above.

```
\protected@edef\Shortb{\@nameauth@Root{#2}}%
\protected@edef\Shortd{\@nameauth@Root{#4}}%
\protected@edef\CapShort{\@nameauth@CapRoot{#4}}%
\protected@edef\AllCapShort{\@nameauth@AllCapRoot{#4}}%
\ifx\testb\testbr
  \let\SNNb\Shortb%
  \protected@edef\Suffb{\@empty}%
\else
  \protected@edef\Suffb{\@nameauth@Suffix{#2}}%
  \protected@edef\SNNb{\Shortb\ISpace\Suffb}%
\fi
\ifx\testd\testdr
  \protected@edef\Suffd{\@empty}%
  \let\ISNNd\Shortd%
  \let\Reversed\Shortd%
  \let\SNNd\Shortd%
  \if@nameauth@DoCaps
    \let\SNNd\CapShort%
    \let\Reversed\CapShort%
  \fi
  \if@nameauth@AllThis
    \let\SNNd\AllCapShort%
    \let\Reversed\AllCapShort%
  \fi
\else
  \protected@edef\Suffd{\@nameauth@Suffix{#4}}%
  \protected@edef\ISNNd{\Shortd\ISpace\Suffd}%
  \protected@edef\Reversed{\Suffd\Space\Shortd}%
  \protected@edef\SNNd{\Shortd\Space\Suffd}%
  \if@nameauth@DoCaps
    \protected@edef\Reversed{\Suffd\Space\CapShort}%
    \protected@edef\SNNd{\CapShort\Space\Suffd}%
  \fi
  \if@nameauth@AllThis
    \protected@edef\Reversed{\Suffd\Space\AllCapShort}%
    \protected@edef\SNNd{\AllCapShort\Space\Suffd}%
  \fi
\fi
```

Here we parse names.

```
\ifx\testc\@empty
  \ifx\teste\@empty
```

For mononyms and name/suffix pairs: If a pseudonym has not been generated by `\AKA` or `\ExcludeName`, and if the proposed pseudonym is not already a mainmatter or frontmatter name, then generate a *see* reference from the pseudonym to a name that will appear in the index.

```
    \ifcsname\csd!PN\endcsname
      \PackageWarning{nameauth}%
      {macro \AKA: XRef: #4 exists}%
    \else
      \ifcsname\csd!MN\endcsname
        \PackageWarning{nameauth}%
        {macro \AKA: Name reference: #4 exists; no xref}%
      \else
        \ifcsname\csd!NF\endcsname
          \PackageWarning{nameauth}%
          {macro \AKA: Name reference: #4 exists; no xref}%
        \else
          \csgdef{\csd!PN}{}%
          \ifx\testa\@empty
            \@nameauth@Index{\csd}%
              {\ISNNd|see{\SNNb}}%
          \else
            \ifx\Suffb\@empty
              \@nameauth@Index{\csd}%
                {\ISNNd|see{\SNNb,\space\arga}}%
            \else
              \@nameauth@Index{\csd}%
                {\ISNNd|see{\Shortb,\space\arga,\space\Suffb}}%
            \fi
          \fi
        \fi
      \fi
    \fi
```

Print an appropriate version of the pseudonym (capped, reversed, etc.) in the text with no special formatting even if no cross-reference was generated in the index. Again, `\expandafter` is used for the punctuation detection.

```
    \if@nameauth@RevThisComma
      \protected@edef\ISpace{,\space}%
      \protected@edef\Space{,\space}%
      \if@nameauth@NBSP
        \protected@edef\Space{,\nobreakspace}%
      \fi
    \fi
    \if@nameauth@RevThis
      \ex\@nameauth@FmtName\ex{\Reversed}%
    \else
      \ex\@nameauth@FmtName\ex{\SNNd}%
    \fi
  \else
```

For name/affix using the old syntax: If a pseudonym has not been generated by `\AKA` or `\ExcludeName`, and if the proposed pseudonym is not already a mainmatter or frontmatter name, then generate a *see* reference from the pseudonym to a name that will appear in the index.

```
      \ifcsname\csde!PN\endcsname
        \PackageWarning{nameauth}%
        {macro \AKA: XRef: #4 #5 exists}%
      \else
        \ifcsname\csde!MN\endcsname
          \PackageWarning{nameauth}%
          {macro \AKA: Name reference: #4 #5 exists; no xref}%
        \else
          \ifcsname\csde!NF\endcsname
            \PackageWarning{nameauth}%
            {macro \AKA: Name reference: #4 #5 exists; no xref}%
          \else
            \csgdef{\csde!PN}{}%
            \ifx\testa\@empty
              \@nameauth@Index{\csde}%
                {\ISNNd\ISpace\arge|see{\SNNb}}%
            \else
              \ifx\Suffb\@empty
                \@nameauth@Index{\csde}%
                  {\ISNNd\ISpace\arge|see{\SNNb,\space\arga}}%
              \else
                \@nameauth@Index{\csde}%
                {\ISNNd\ISpace\arge|see{\Shortb,\space\arga,\space\Suffb}}%
              \fi
            \fi
          \fi
        \fi
      \fi
```

Print an appropriate version of the pseudonym (capped, reversed, etc.) in the text with no special formatting even if no cross-reference was generated in the index.

```
      \if@nameauth@RevThisComma
        \protected@edef\ISpace{,\space}%
        \protected@edef\Space{,\space}%
        \if@nameauth@NBSP
          \protected@edef\Space{,\nobreakspace}%
        \fi
      \fi
      \if@nameauth@AltAKA
        \ex\@nameauth@FmtName\ex{\arge}%
      \else
        \if@nameauth@RevThis
          \ex\@nameauth@FmtName\ex{\ex\arge\ex\Space\SNNd}%
        \else
          \ex\@nameauth@FmtName\ex{\ex\SNNd\ex\space\arge}%
        \fi
      \fi
    \fi
  \else
```

For Western names and affixes: If a pseudonym has not been generated by `\AKA` or `\ExcludeName`, and if the proposed pseudonym is not already a mainmatter or frontmatter name, then generate a *see* reference from the pseudonym to a name that will appear

in the index.

```
    \ifcsname\cscd!PN\endcsname
      \PackageWarning{nameauth}%
      {macro \AKA: XRef: #3 #4 exists}%
    \else
      \ifcsname\cscd!MN\endcsname
        \PackageWarning{nameauth}%
        {macro \AKA: Name reference: #3 #4 exists; no xref}%
      \else
        \ifcsname\cscd!NF\endcsname
          \PackageWarning{nameauth}%
          {macro \AKA: Name reference: #3 #4 exists; no xref}%
        \else
          \csgdef{\cscd!PN}{}%
          \ifx\testa\@empty
            \ifx\Suffd\@empty
              \@nameauth@Index{\cscd}%
                {\ISNNd,\space\argc|see{\SNNb}}%
            \else
              \@nameauth@Index{\cscd}%
                {\Shortd,\space\argc,\space\Suffd|see{\SNNb}}%
            \fi
          \else
            \ifx\Suffb\@empty
              \ifx\Suffd\@empty
                \@nameauth@Index{\cscd}%
                  {\ISNNd,\space\argc|see{\SNNb,\space\arga}}%
              \else
                \@nameauth@Index{\cscd}%
                {\Shortd,\space\argc,\space\Suffd|see{\SNNb,\space\arga}}%
              \fi
            \else
              \ifx\Suffd\@empty
                \@nameauth@Index{\cscd}%
                {\ISNNd,\space\argc|see{\Shortb,\space\arga,\space\Suffb}}%
              \else
                \@nameauth@Index{\cscd}%
{\Shortd,\space\argc,\space\Suffd|see{\Shortb,\space\arga,\space\Suffb}}%
              \fi
            \fi
          \fi
        \fi
      \fi
    \fi
```

Print an appropriate version of the pseudonym (capped, reversed, etc.) in the text with no special formatting even if no cross-reference was generated in the index.

```
    \if@nameauth@RevThisComma
      \protected@edef\ISpace{,\space}%
      \protected@edef\Space{,\space}%
      \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
    \fi
    \ifx\teste\@empty
      \let\FNN\argc%
    \else
      \let\FNN\arge%
    \fi
    \if@nameauth@RevThis
      \ex\@nameauth@FmtName\ex{\ex\SNNd\ex\Space\FNN}%
    \else
      \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNNd}%
    \fi
  \fi
```

Reset all the "per name" Boolean values.

```
  \@nameauth@NBSPfalse%
  \@nameauth@AltAKAfalse%
  \@nameauth@DoCapsfalse%
  \@nameauth@AllThisfalse%
  \@nameauth@ShowCommafalse%
  \@nameauth@RevThisfalse%
  \@nameauth@RevThisCommafalse%
```

Call the full stop detection.

```
  \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
}
```

`\AKA*` This starred form sets a Boolean to print only the alternate name parameter, if that exists, and calls `\AKA`.

```
\WithSuffix\def\AKA*{\@nameauth@AltAKAtrue\AKA}
```

`\PName` `\PName` is a convenience macro that calls `\Name`, then `\AKA`.

```
\newcommandx*\PName[5][1=\@empty,3=\@empty,5=\@empty]%
{%
  \Name[#1]{#2}\space(\AKA[#1]{#2}[#3]{#4}[#5])%
}
```

`\PName*` This just calls `\Name*`, then `\AKA`.

```
\WithSuffix\def\PName*{\@nameauth@FullNametrue\PName}
```

\TagName This creates an index entry tag that is applied to a name that is not already used as a cross reference via \AKA.

```
\newcommandx*\TagName[4][1=\@empty, 3=\@empty]%
{%
  \protected@edef\testa{#1}%
  \protected@edef\testb{\trim@spaces{#2}}%
  \protected@edef\testc{#3}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and then we parse the arguments, defining the tag control sequences.

```
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \TagName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \TagName: Essential name malformed}%
    \fi
  \fi
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!PN\endcsname
        \PackageWarning{nameauth}%
        {macro \TagName: not tagging xref: #2}%
      \else
        \csgdef{\csb!TAG}{#4}%
      \fi
    \else
      \ifcsname\csbc!PN\endcsname
        \PackageWarning{nameauth}%
        {macro \TagName: not tagging xref: #2 #3}%
      \else
        \csgdef{\csbc!TAG}{#4}%
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \PackageWarning{nameauth}%
      {macro \TagName: not tagging xref: #1 #2}%
    \else
      \csgdef{\csab!TAG}{#4}%
    \fi
  \fi
}
```

\UntagName This deletes an index entry tag.

```
\newcommandx*\UntagName[3][1=\@empty, 3=\@empty]%
{%
  \protected@edef\testa{#1}%
  \protected@edef\testb{\trim@spaces{#2}}%
  \protected@edef\testc{#3}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and then we parse the arguments, undefining the tag control sequences.

```
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \UntagName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \UntagName: Essential name malformed}%
    \fi
  \fi
  \ifx\testa\@empty
    \ifx\testc\@empty
      \global\csundef{\csb!TAG}%
    \else
      \global\csundef{\csbc!TAG}%
    \fi
  \else
    \global\csundef{\csab!TAG}%
  \fi
}
```

\PretagName This creates an index entry tag that is applied before a name.

```
\newcommandx*\PretagName[4][1=\@empty, 3=\@empty]%
{%
  \protected@edef\testa{#1}%
  \protected@edef\testb{\trim@spaces{#2}}%
  \protected@edef\testc{#3}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and then we parse the arguments, defining the tag control sequences.

```
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \TagName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \TagName: Essential name malformed}%
    \fi
  \fi
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!PN\endcsname
        \PackageWarning{nameauth}%
        {macro \PretagName: tagging xref: #2}%
      \fi
      \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
    \else
      \ifcsname\csbc!PN\endcsname
        \PackageWarning{nameauth}%
        {macro \PretagName: tagging xref: #2 #3}%
      \fi
      \if@nameauth@Pretag\csgdef{\csbc!PRE}{#4\@nameauth@Actual}\fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \PackageWarning{nameauth}%
      {macro \PretagName: tagging xref: #1 #2}%
    \fi
    \if@nameauth@Pretag\csgdef{\csab!PRE}{#4\@nameauth@Actual}\fi
  \fi
}
```

`\IndexName` This creates an index entry that is not already a pseudonym. It prints nothing. It does ensure consistent formatting.

```
\newcommandx*\IndexName[3][1=\@empty, 3=\@empty]%
{%
  \protected@edef\testa{#1}%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\testb{\trim@spaces{#2}}%
  \protected@edef\testbr{\@nameauth@Root{#2}}%
  \protected@edef\testc{#3}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and make parsing decisions. Below we handle the types of spaces or commas that will be inserted into the index entries.

```
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \IndexName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \IndexName: Essential name malformed}%
    \fi
  \fi
  \protected@edef\Space{\space}%
  \if@nameauth@AlwaysComma
    \protected@edef\Space{,\space}%
  \fi
  \if@nameauth@ShowComma
    \protected@edef\Space{,\space}%
  \fi
```

Now we deal with suffixes, and whether to handle them for Western or Eastern names.

```
  \let\Short\testbr%
  \ifx\testb\testbr
    \let\SNN\Short%
    \protected@edef\Suff{\@empty}%
  \else
    \protected@edef\Suff{\@nameauth@Suffix{#2}}%
    \protected@edef\SNN{\Short\Space\Suff}%
  \fi
```

We create the appropriate index entries with tags, letting the internal indexing macro sort that out. We do not create an index entry in the case that a name has been used as a pseudonym by `\AKA` or `\ExcludeName`.

```
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!PN\endcsname
        \PackageWarning{nameauth}%
        {macro \IndexName: XRef: #2 exists}%
      \else
        \@nameauth@Index{\csb}{\SNN}%
      \fi
    \else
      \ifcsname\csbc!PN\endcsname
        \PackageWarning{nameauth}%
        {macro \IndexName: XRef: #2 #3 exists}%
      \else
        \@nameauth@Index{\csbc}{\SNN\Space\argc}%
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \PackageWarning{nameauth}%
      {macro \IndexName: XRef: #1 #2 exists}%
    \else
      \ifx\Suff\@empty
        \@nameauth@Index{\csab}{\Short,\space\arga}%
      \else
        \@nameauth@Index{\csab}{\Short,\space\arga,\space\Suff}%
      \fi
    \fi
  \fi
  \@nameauth@ShowCommafalse%
}
```

`\ExcludeName` This macro prevents a name from being formatted or indexed, making `\Name` and friends print their arguments, emit a warning, and continue.

```
\newcommandx*\ExcludeName[3][1=\@empty, 3=\@empty]%
{%
  \protected@edef\testa{#1}%
  \protected@edef\testb{\trim@spaces{#2}}%
  \protected@edef\testc{#3}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and make parsing decisions. Below we parse the name arguments and create a pseudonym control sequence if it does not exist.

```
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \ExcludeName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \ExcludeName: Essential name malformed}%
    \fi
  \fi
```

```
\ifx\testa\@empty
  \ifx\testc\@empty
    \ifcsname\csb!PN\endcsname
      \PackageWarning{nameauth}%
      {macro \ExcludeName: Xref: #2 already exists}%
    \else
      \ifcsname\csb!MN\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Reference: #2 exists; no exclusion}%
      \else
        \ifcsname\csb!NF\endcsname
          \PackageWarning{nameauth}%
          {macro \ExcludeName: Reference: #2 exists; no exclusion}%
        \else
          \csgdef{\csb!PN}{}%
        \fi
      \fi
    \fi
  \else
    \ifcsname\csbc!PN\endcsname
      \PackageWarning{nameauth}%
      {macro \ExcludeName: Xref: #2 #3 already exists}%
    \else
      \ifcsname\csbc!MN\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Reference: #2 #3 exists; no exclusion}%
      \else
        \ifcsname\csbc!NF\endcsname
          \PackageWarning{nameauth}%
          {macro \ExcludeName: Reference: #2 #3 exists; no exclusion}%
        \else
          \csgdef{\csbc!PN}{}%
        \fi
      \fi
    \fi
  \fi
\else
  \ifcsname\csab!PN\endcsname
    \PackageWarning{nameauth}%
    {macro \ExcludeName: XRef: #1 #2 already exists}%
  \else
    \ifcsname\csab!MN\endcsname
      \PackageWarning{nameauth}%
      {macro \ExcludeName: Reference: #1 #2 exists; no exclusion}%
    \else
      \ifcsname\csab!NF\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Reference: #1 #2 exists; no exclusion}%
      \else
        \csgdef{\csab!PN}{}%
      \fi
    \fi
  \fi
\fi
}
```

\ForgetName This undefines a control sequence to force the "first use" option of \Name.

```
\newcommandx*\ForgetName[3][1=\@empty, 3=\@empty]%
{%
  \protected@edef\testa{#1}%
  \protected@edef\testb{\trim@spaces{#2}}%
  \protected@edef\testc{#3}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and then we parse the arguments, undefining the control sequences.

```
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \ForgetName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \ForgetName: Essential name malformed}%
    \fi
  \fi
  \ifx\testa\@empty
    \ifx\testc\@empty
      \global\csundef{\csb!MN}%
      \global\csundef{\csb!NF}%
    \else
      \global\csundef{\csbc!MN}%
      \global\csundef{\csbc!NF}%
    \fi
  \else
      \global\csundef{\csab!MN}%
      \global\csundef{\csab!NF}%
  \fi
}
```

\SubvertName This defines a control sequence to suppress the "first use" of \Name.

```
\newcommandx*\SubvertName[3][1=\@empty, 3=\@empty]%
{%
  \protected@edef\testa{#1}%
  \protected@edef\testb{\trim@spaces{#2}}%
  \protected@edef\testc{#3}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We make copies of the arguments to test them and then we parse the arguments, defining the control sequences.

```
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \SubvertName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \SubvertName: Essential name malformed}%
    \fi
  \fi
  \ifx\testa\@empty
    \ifx\testc\@empty
      \csgdef{\csb!MN}{}%
      \csgdef{\csb!NF}{}%
    \else
      \csgdef{\csbc!MN}{}%
      \csgdef{\csbc!NF}{}%
    \fi
  \else
      \csgdef{\csab!MN}{}%
      \csgdef{\csab!NF}{}%
  \fi
}
```

nameauth The nameauth environment provides a means to implement shorthand references to names in a document.

```
\newenvironment{nameauth}{%
  \begingroup%
  \let\ex\expandafter%
  \csdef{<}##1&##2&##3&##4>{%
    \protected@edef\arga{\trim@spaces{##1}}%
    \protected@edef\testb{\trim@spaces{##2}}%
    \protected@edef\testc{\trim@spaces{##3}}%
    \protected@edef\testd{\trim@spaces{##4}}%
    \newtoks\tokb%
    \newtoks\tokc%
    \newtoks\tokd%
    \tokb\expandafter{##2}%
    \tokc\expandafter{##3}%
    \tokd\expandafter{##4}%
    \ifx\arga\@empty
        \PackageError{nameauth}%
        {environment nameauth: Control sequence missing}%
    \else
      \ifx\testc\@empty
        \PackageError{nameauth}%
        {environment nameauth: Essential name missing}%
      \else
        \ifcsname\arga\endcsname
          \PackageWarning{nameauth}%
          {environment nameauth: Redefinition of shorthands}%
        \fi
        \ifx\testd\@empty
          \ifx\testb\@empty
            \ex\csxdef\ex{\ex\arga\ex}\ex{\ex\Name\ex{\the\tokc}}%
            \ex\csgdef\ex{\ex L\ex\arga\ex}\ex{\ex\Name\ex*\ex{\the\tokc}}%
            \ex\csgdef\ex{\ex S\ex\arga\ex}\ex{\ex\FName\ex{\the\tokc}}%
          \else
            \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\arga\ex\ex\ex}%
              \ex\ex\ex{\ex\ex\ex\Name\ex\ex\ex[\ex\the\ex\tokb\ex]%
              \ex{\the\tokc}}%
            \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\arga%
              \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\Name\ex\ex\ex*%
              \ex\ex\ex[\ex\the\ex\tokb\ex]\ex{\the\tokc}}%
            \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\arga%
              \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\FName\ex\ex\ex[%
              \ex\the\ex\tokb\ex]\ex{\the\tokc}}%
          \fi
        \else
          \ifx\testb\@empty
            \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\arga\ex\ex\ex}%
              \ex\ex\ex{\ex\ex\ex\Name\ex\ex\ex{\ex\the\ex\tokc\ex}%
              \ex[\the\tokd]}%
            \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex L\ex\ex\ex\arga%
              \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\Name%
              \ex\ex\ex*\ex\ex\ex{\ex\the\ex\tokc\ex}\ex[\the\tokd]}%
            \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex S\ex\ex\ex\arga%
              \ex\ex\ex}\ex\ex\ex{\ex\ex\ex\FName\ex\ex\ex{%
              \ex\the\ex\tokc\ex}\ex[\the\tokd]}%
          \else
```

```
            \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
              \ex\ex\ex\ex\ex\ex\ex\arga\ex\ex\ex\ex\ex\ex\ex}%
              \ex\ex\ex\ex\ex\ex\ex{\ex\ex\ex\ex\ex\ex\ex\Name%
              \ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex\tokb%
              \ex\ex\ex]\ex\ex\ex{\ex\the\ex\tokc\ex}\ex[\the\tokd]}%
            \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
              \ex\ex\ex\ex\ex\ex\ex L\ex\ex\ex\ex\ex\ex\ex\arga%
              \ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex{%
              \ex\ex\ex\ex\ex\ex\ex\Name\ex\ex\ex\ex\ex\ex\ex*%
              \ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex\tokb%
              \ex\ex\ex]\ex\ex\ex{\ex\the\ex\tokc\ex}\ex[\the\tokd]}%
            \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
              \ex\ex\ex\ex\ex\ex\ex S\ex\ex\ex\ex\ex\ex\ex\arga%
              \ex\ex\ex\ex\ex\ex\ex}\ex\ex\ex\ex\ex\ex\ex{%
              \ex\ex\ex\ex\ex\ex\ex\FName\ex\ex\ex\ex\ex\ex\ex[%
              \ex\ex\ex\the\ex\ex\ex\tokb\ex\ex\ex]%
              \ex\ex\ex{\ex\the\ex\tokc\ex}\ex[\the\tokd]}%
          \fi
        \fi
      \fi
    \fi
    \ignorespaces%
  }\ignorespaces%
}{\endgroup\ignorespaces}
```

4 Change History

5 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.