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

Charles P. Schaum†

Released 2016/01/05

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 to be compatible with A4 and US letter. Macro references are minimized for a "clean" index, showing how nameauth handles indexing.

*This file describes version v2.3, last revised 2016/01/05.

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

1 Introduction

1.1 Preliminaries

When publications use hundreds of names, it takes time and money to check them. This package automates much of that work. **Context determines name forms** unless otherwise modified, meaning that **you usually do not have to retype names** when editing a document. You can **implement a name authority** that allows for name variants in the text and consistent index entries. With nameauth you can handle some **cross-cultural naming conventions.** Additionally, you can use **index sort keys and tags** automatically after assigning them.

This package grew from generalized needs for translating old German and Latin texts. Design principles include:

1. Format and vary name forms according to standard syntax in the body text, independent of the index.
    - Default to long name references first, then shorter ones.
    - Use alternate names only in the body text, not the index.
    - Perform name caps and reversing only in the body text.
2. Perform typographic formatting of names only in the body text. Reflect source text typography with English conventions, but only *after* syntactic formatting is complete.
3. Allow typographic formatting to be customized and permit control sequences in names (Sections 2.5.7, 2.5.8, and 2.8.4) to allow Continental and non-English standards.
4. Always aim to reduce keystrokes.
5. Accommodate the broadest set of names while minimizing keystrokes.

This manual performs something of a "torture test" on this package. You might want to avoid doing that if you are a beginner.

Nameauth depends on etoolbox, ifxetex, ifluatex, suffix, trimspaces, and xargs. It was tested with `latex`, `lualatex`, `pdflatex`, and `xelatex`, along with `makeindex` and `texindy`. This manual was typeset with `pdflatex` using `makeindex` and `gind.ist`.

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. At no time do I intend any disrespect or statement of bias regarding any particular person, culture, or tradition. All names are used only for teaching purposes.

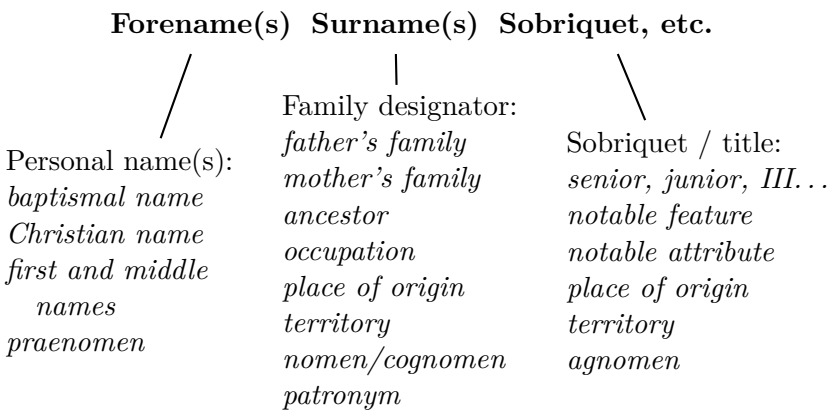
1.2 What's In A Name?

Name forms are ambiguous apart from historical and cultural contexts. The nameauth package helps you encode names from as many contexts as possible.

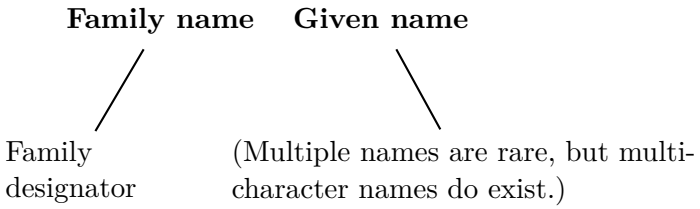
In this manual we refer to three classes of names. A "surnames" argument, ⟨*SNN*⟩, denotes a "required name," that is, a Western surname, an Eastern family name, or an ancient/medieval name.[1] Other naming systems can be adapted to these categories, *e.g.*, Icelandic, Hungarian, etc.

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.

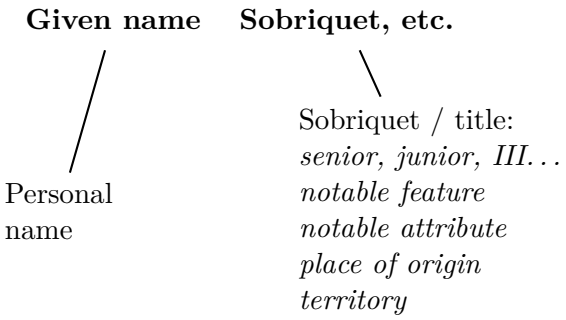
1. Western name:



2. Eastern name:



3. Ancient name:



[1] Some professional literature speaks of forenames and optional surnames. See Mulvany, *Indexing Books*, pages 152–82, which I used as a guide along with the *Chicago Manual of Style*. That approach does not work in LaTeX, where we use optional forenames for the same effect.

2 Usage

2.1 Package Options

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

From the user's perspective these options proceed from the most general to more specific details. Package options address the following:

1. Enable or disable features (formatting, indexing, index sorting)
2. Affect the syntax of names (commas, capitalization, and reversing)
3. Typographic display of names (formatted or not, and how)

Default options are in boldface.

Enable/Disable Features

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 `mainmatter` option 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, but they implement two completely separate systems of first/subsequent names. These systems manage names in separate contexts. See Section 2.5.9.

Enable/Disable Indexing

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

The default `index` option enables name indexing right away. The `noindex` option disables the indexing of names until `\IndexActive` enables it. That can affect the use of index tags. This applies only to naming and indexing macros in the `nameauth` package. See Section 2.8.1.

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 2.8.4.

Affect the Syntax of Names

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:

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

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. One of the design principles of this package keeps it consistent with English typography and syntax. Thus no syntactic or typographic changes are propagated into the index by default.

Still, you can use this package with different conventions that involve both syntax and formatting. You can type in capitalized family names directly to get that effect. See also Section 2.5.8 on how to use macros to get caps (`\uppercase`) or small caps (`\textsc`) in both the body text and the index. This becomes easy with the simplified interface.

Section 2.4.2 deals with 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 "Surname, Forenames" order, meant for Western names.

See Section 2.4.2 for related macros to control name reversing by section or by name. This also affects how Eastern names will appear in the index.

So-called "last-comma-first" lists of names via `allrevcomma` (Section 2.5.4) are *not* the same as the `comma` option. They are designed for Western names.

[2] Before version 0.9 the nameauth package assumed the `comma` option by default and used the old syntax to encode names. Newer versions are backward-compatible.

Typographic Display of Names

Section 2.5.8 explains in greater detail that typographic display is different from the syntactic formatting of names and occurs after syntactic formatting is complete. This package is designed with type hierarchies in mind. See Robert Bringhurst, *The Elements of Typographic Style*, version 3.2 (Point Roberts, Washington: Hartley & Marks, 2008), 53–60.

Even though English typography was the design choice due to native access (for me) and global reach (thanks, England), a stint in Germany helped to make me aware of continental typography. Especially in recent versions, one can adapt this package for use in Continental and other typographic standards, as Section 2.5.8 discusses. One must use sort tags, as Section 2.8.4 explains.

Continental standards require you to know the surname in advance and draw formatting into the syntactic elements of the surname. By default, `nameauth` uses a "serendipitous ambiguity" of ancient, Eastern, and suffixed name forms handled by the ⟨*Surname, affix*⟩ pattern that is resolved subtly by several factors. I burden Continental users with a few more keystrokes in order to minimize keystrokes among the broadest group of users and names.

If you intend to use this package for Continental publishing where the surname is formatted with small caps in the running text and the index, use the `noformat` option. Otherwise, the options below are meant generally for applications in English typography. The default is `smallcaps` because this package was developed to aid my editing and translation of older German and Latin documents into English. I do apologize for any inconvenience in design choices.

Formatting Attributes

`alwaysformat`	If formatting is enabled by the `mainmatter` option or by `\NamesActive`, this option causes all names to have typographic formatting applied to them, whether first or subsequent uses.
`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.

Section 2.5.8 offers even more possibilities for presenting the first use of names. That includes typographic formatting that also adds names in margins.[3] This allows one to implement some creative solutions for presenting names.

[3] Two books that helped my thinking about such typographic concepts include Bernhard Lohse, *Luthers Theologie* (Göttingen: Vandenhoeck & Ruprecht, 1995) and the five-volume series by Jaroslav J. Pelikan Jr., The Christian Tradition: A History of the Development of Doctrine (Chicago: Chicago UP, 1971–89). Each volume in the series has its own title.

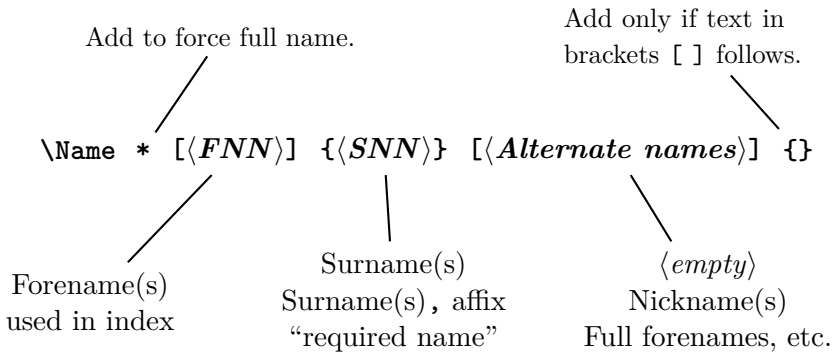
Given the use of small caps in the text and index in Lohse's volume, I looked at several possibilities and arrived at the current solution for Continental users. Pelikan's volumes pointed me to the idea of a typographic first use of names that could well be served by a margin paragraph. Being a history enthusiast, I implemented that design idea in this manual.

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 macro arguments ⟨*forename(s)*⟩ with ⟨*FNN*⟩ and ⟨*surname(s)*⟩ with ⟨*SNN*⟩. Use the `nocomma` option especially when using Eastern names and ancient names.

Western Names



Usual forms:

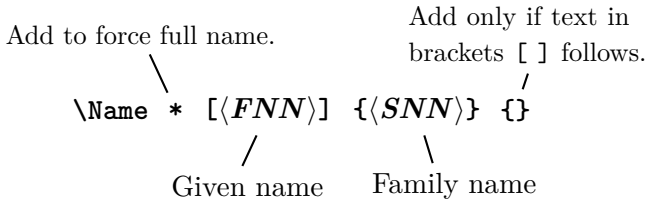
```
\Name[⟨FNN⟩]{⟨SNN⟩}                    \Name[George]{Washington}
\Name[⟨FNN⟩]{⟨SNN, affix⟩}    \Name[John David]{Rockefeller, II}
```

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

```
\Name[⟨FNN⟩]{⟨SNN⟩}[⟨Alternate names⟩]
                                  \Name[Bob]{Hope}[Leslie Townes]
                                \Name[Clive Staples]{Lewis}[C.S.]
\Name[⟨FNN⟩]{⟨SNN, affix⟩}[⟨Alternate names⟩]
                              \Name[John David]{Rockefeller, IV}[Jay]
```

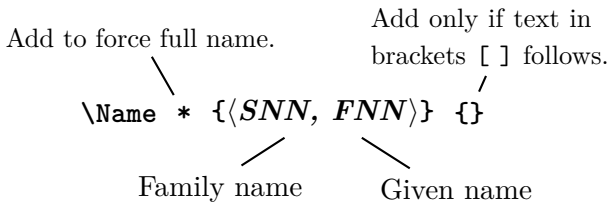
The older syntax is `\Name{⟨SNN⟩}[⟨affix⟩]`. See Section 2.2.3 for its usage and its shortcomings. It remains for backward compatibility.

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 both comma-delimited forms and the old syntax.

Eastern Names in the Text, Eastern-style Index



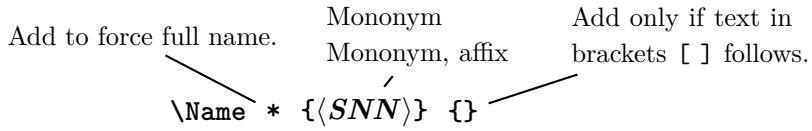
Usual form:

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

These names truly are Eastern names. They 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" Eastern form.

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}`
`\Name{Attila, the Hun}`

These forms are used for royalty, ancient figures, and other monomyms with or without suffixes.[4] The older syntax takes the form `\Name{`⟨*Mononym*⟩`}[`⟨*affix*⟩`]`. Cf. Section 2.2.3.

The main interface allows one to use `\Name`⟨*arguments*⟩ as a base for many other package macros. For example, all these are variants of a base pattern, where ⟨*prefix macro*⟩ consist of one or more of `\CapThis`, `\CapName`, `\RevName`, `\RevComma`, `\ShowComma`, and `\KeepAffix`:

⟨*prefix macro*⟩	`\Name`	⟨*arguments*⟩	
⟨*prefix macro*⟩	`\Name*`	⟨*arguments*⟩	
⟨*prefix macro*⟩	`\FName`	⟨*arguments*⟩	
	`\IndexName`	⟨*arguments*⟩	
	`\ForgetName`	⟨*arguments*⟩	
	`\SubvertName`	⟨*arguments*⟩	
	`\PretagName`	⟨*arguments*⟩	⟨*sort key*⟩
	`\TagName`	⟨*arguments*⟩	⟨*tag*⟩
	`\UntagName`	⟨*arguments*⟩	
	`\ExcludeName`	⟨*arguments*⟩	

[4] 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.

2.2.2 Simplefied Interface

`nameauth` The `nameauth` environment allows one to save typing and aid consistency by using shorthands. It replaces the use of `\Name`, `\Name*`, and `\FName`, but not the other naming macros. Thus one must remain aware of the main interface.

The simplified interface 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.[5] The italicized comments at right are not part of the example proper, but are there for explanation. Macro fields have uniform widths only to help compare argument types.

```
\begin{nameauth}
  \< ⟨cseq1⟩ & ⟨FNN⟩ & ⟨SNN⟩                & >                    Western[6]
  \< ⟨cseq2⟩ & ⟨FNN⟩ & ⟨SNN, affix⟩ & >                              Western
  \< ⟨cseq3⟩ & ⟨FNN⟩ & ⟨SNN⟩                & ⟨Alt. names⟩ >      W. nickname[7]
  \< ⟨cseq4⟩ & ⟨FNN⟩ & ⟨SNN, affix⟩ & ⟨Alt. names⟩ >              W. nickname
  \< ⟨cseq5⟩ &        & ⟨SNN⟩                & >                    ancient/mono
  \< ⟨cseq6⟩ &        & ⟨SNN, affix⟩ & >                           royal/ancient
  \< ⟨cseq7⟩ &        & ⟨SNN, FNN⟩ & >                             Eastern[8]
  \< ⟨cseq8⟩ &        & ⟨SNN⟩                & ⟨FNN/affix⟩ >       old syntax[9]
\end{nameauth}
```

Each ⟨*cseq*⟩ creates three macros. In the document text, `\`⟨*cseq*⟩ itself works like `\Name`. `\L`⟨*cseq*⟩ (think "Long") works like `\Name*`. `\S`⟨*cseq*⟩ (think "Short") works like `\FName`. Please bear in mind the following guidelines:

- In this context, "`\<`" is an escape character and a control sequence. If you forget it or just use `<` without the backslash, you will get errors.
- There *must* be four argument fields (three ampersands) per line. Leaving out an ampersand will cause an error. Think "holy hand grenade of Antioch" from *Monty Python and the Holy Grail*.
- Leading and trailing spaces in each `&`-delimited field are stripped, as is also the case in the main interface.
- As in the main interface, medial spaces do not affect first-use control sequences, but they will affect name forms in the body text and index.
- In the document text, as with the main interface, include trailing braces `{}`, control spaces, or the like if text in brackets `[]` follows any of the shorthands, *e.g.*, `\LWash{} [\emph{sic}]`.
- The old syntax (Section 2.2.3), triggered by an empty ⟨*FNN*⟩ field, causes the ⟨*Alt. names*⟩ field to be interpreted as either Eastern ⟨*FNN*⟩ or an ⟨*affix*⟩. Due to its limitations and potential confusion, you are encouraged to avoid it unless you are using the `comma` option.

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

[6] This is also the form used with "non-native" Eastern names using reversing macros, but leaving them in Western form in the index.

[7] When the ⟨*Alt. names*⟩ is used, ⟨*FNN*⟩ never appears in the body text, but only in the index. See Section 2.3.2 to avoid possible difficulties. You could use `\AKA` to create a *see* reference for the Jay Rockefeller example on the next page; see Section 2.7.1.

[8] "Native" Eastern names can be reversed to use Western order in the body text, but they will always have an Eastern form in the index.

[9] This is the old syntax for Eastern and royal names.

The example below illustrates a fairly complete set of names, apart from some special cases covered elsewhere in the manual:

```
\begin{nameauth}
  \< Wash & George & Washington & >                    Western
  \< Soto & Hernando & de Soto & >                     particle
  \< JRII & John David & Rockefeller, II & >           affix
  \< JRIV & John David & Rockefeller, IV & >           affix
  \< JayR & John David & Rockefeller, IV & Jay >      nickname
  \< Lewis & Clive Staples & Lewis & C.S. >           nickname
  \< Jack & Clive Staples & Lewis & Jack >           nickname
  \< Aris & & Aristotle & >                           ancient
  \< Eliz & & Elizabeth, I & >                       royal
  \< Attil & & Attila, the Hun & >                   ancient
  \< Konoe & Fumimaro & Konoe & >          Eastern/Western index
  \< Yamt & & Yamamoto, Isoroku & >        Eastern/Eastern index
\end{nameauth}
```

Now we see how this works in the body text, which you can compare with the index. A dagger (†) indicates an Eastern name with a Western index form.

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:

`\JRII`	John David Rockefeller II
`\LJRII`	John David Rockefeller II
`\JRII`	Rockefeller

Nicknames: (See Section 2.3.2)

`\JRIV`	John David Rockefeller IV
`\LJRIV[Jay]`	Jay Rockefeller IV
`\SJRIV[Jay]`	Jay
`\SJRIV[Jay] \JRIV`	Jay Rockefeller
`\LJayR`	Jay Rockefeller IV
`\SJayR`	Jay
`\SJayR\ \JayR`	Jay Rockefeller
`\Lewis`	C.S. Lewis
`\Lewis`	Lewis
`\LJack`	Jack Lewis
`\SJack`	Jack

Ancient:

`\Aris`	Aristotle
`\Aris`	Aristotle

Medieval/Royal:

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

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

Sections 2.5.1, 2.5.7, and 2.8.4 deal with the pitfalls of accents and capitalization, as well as why you should use `\PretagName` for any name with control sequences or extended Unicode under NFSS. This becomes very important when authors and publishers use medieval names as Western names.

When tagging or pre-tagging names (Section 2.8.4), the ⟨*Alternate names*⟩ field has no effect on index tags. `\JRIV` and `\JayR` need only one tag:

`\TagName[John David]{Rockefeller, IV}{`⟨*something*⟩`}`

Likewise, `\Lewis` and `\Jack` need only one tag:

`\TagName[Clive Staples]{Lewis}{`⟨*something*⟩`}`

Only in the ⟨*Alternate names*⟩ field can one use control sequences without them affecting index sorting, unlike in the other fields. If the ⟨*FNN*⟩ field is empty, then you have the situation below.

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 to avoid possible confusion and error.

The `comma` option causes Western names with affixes to have a comma. Yet that also causes Eastern and ancient names, or any names using a pattern like ⟨*SNN, affix*⟩ or ⟨*SNN, FNN*⟩ to display a comma where it should not occur. In that case the older syntax is needed. The old form lacks some error checking and robustness contained in the new syntax and limits the use of several macros, including `\AKA`. Section 2.11 offers some cautions about the old syntax, as do many places in this manual. The form is:

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

Here the ⟨*FNN*⟩ fields are empty. That changes the final field from ⟨*Alternate names*⟩ to ⟨*affix/Eastern FNN*⟩.

`\Dagb` gives Dagobert I, then Dagobert. In similar fashion, 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.[10]

Avoid `\Name{Henry, VIII}[Tudor]` unless you want "Henry VIII Tudor" and "Henry" in the body text and "Henry VIII Tudor" in the index. One solution adds "Tudor" as needed in the text after `\Name{Henry, VIII}` and uses a tag in the index: `\TagName{Henry, VIII}{ Tudor}` (see Section 2.8.5).

[10] Technically you can mix the two, as I do here. You must force first and subsequent uses with `\ForgetName` and `\SubvertName`. You must 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.

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 body text only. It does this if the ⟨*FNN*⟩ field is not empty; see "Cicero" above. Regarding their index entries, `\Name[M.T.]{Cicero}[Marcus Tullius]` and `\Name[M.T.]{Cicero}` are equivalent. This lets one use a nickname while keeping the indexed form constant. If the ⟨*FNN*⟩ is empty, you get the old syntax for Eastern and royal names (Section 2.2.3).

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

Here we have the equivalent with the simplified interface. `\Einstein`, `\LEinstein`, and `\Einstein` produce Albert Einstein, Albert Einstein, and Einstein. `\CBald` and `\CBald` give Charles the Bald and Charles. `\Confucius` yields Confucius and Confucius. `\Cicero` prints M.T. Cicero and Cicero, while `\LCicero[Marcus Tullius]` gives Marcus Tullius Cicero. The next page explains why this form may be preferable in some cases for name variants when using the simplified interface.

2.3.2 Forenames: \FName

\FName
\FName*

\FName and its synonym \FName* print just forenames, but only in subsequent name uses.[11] They are intended for Western-style names. The syntax is:

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

This macro always prints full name when a name is first used. That prevents a first-name reference before a person has been introduced. To force a short name as a first reference, you could use a macro to incorporate:

```
\SubvertName[⟨FNN⟩]{⟨SNN⟩}%
\makeatletter\@nameauth@FirstFormattrue\makeatother%
\FName[⟨FNN⟩]{⟨SNN⟩}
```

By design, \FName *never* prints Eastern personal names so that ancient names also work (cf. Section 2.10). Examples of general use include:

\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

With the simplified interface example from the previous page, \SEinstein, \SConfucius, \SCicero, and \SCBald give us Albert, Confucius, M.T., and Charles. \SCicero[Marcus Tullius] gives Marcus Tullius. However, with the macro \FName[Chesley B.]{Sullenberger, III}[Sully] we have "Sully Sullenberger III" and "Sully." Please use caution!

This may not always be a "bug." Remembering Section 2.2.2, you can use C.S. Lewis or "Jack." \FName[Clive Staples]{Lewis}[C.S.] or \Lewis gives the first form, while \FName[Clive Staples]{Lewis}[Jack] or \Jack gives the second. \SJayR gave Jay Rockefeller IV and Jay, but the index entry remains "Rockefeller, John David, IV." Using "default nicknames" in the simplified interface has some caveats:

```
\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.]." The ⟨*Alternate names*⟩ field is always occupied when using \Sully, etc. Thus, the final [Chesley B.] is not a macro argument.

[11] The two macros are the same in case you edit \Name* by adding an F to get a first reference, just as you might edit \Name the same way to get the same result.

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 between the comma and affix are ignored. Extra commas have no effect. 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

You cannot use the old syntax with the Hammerstein example. 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.

Compare Older Syntax

Avoid using the older syntax, shown below, except with the `comma` option. The older syntax prevents Eastern and ancient names that use the ⟨*SNN, affix*⟩ pattern from having unwanted commas in them with the `comma` option or with `\ShowComma`. `\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, to avoid potential frustration, please avoid using the older syntax unless you need it.

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. For caps in the text and index see Section 2.5.8.

2.5 Other Naming Topics

Language-Related Issues

2.5.1 Particles

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*.

`\CapThis` In English, these particles go in the ⟨*SNN*⟩ field of `\Name`, *e.g.*, Walter de la Mare. `\CapThis\Name[Walter]{de la Mare}` lets you capitalize *de* when at the beginning of a sentence. De la Mare will think it fair. De Soto (using `\CapThis\Soto` from Section 2.2.2) would agree.

Connect one-character particles with surnames via `~` or `\nobreakspace` to avoid bad breaks and prevent `\CapThis` from eating the space between the particle and the surname. There are no issues when particles have two or more characters in them, as do those listed above.

`\AccentCapThis` `\CapThis` (and `\AccentCapThis`) always work and give the same desired results with `xelatex` and `lualatex`. With `pdflatex` you *must* use `\AccentCapThis` when the first letter of the particle is an extended Unicode character.

```
\PretagName{Thomas, à~Kempis}{Thomas a Kempis}     medieval
\PretagName[Thomas]{à~Kempis}{Thomas a Kempis}     Western
\begin{nameauth}
  \< KempMed & & Thomas, à~Kempis & >              medieval
  \< KempW & Thomas & à~Kempis & >                 Western
\end{nameauth}
```

You do not need either `\CapThis` or `\AccentCapThis` if you use the medieval forms Thomas à Kempis and Thomas. It is important that you do not mix medieval and Western forms to avoid odd name forms and multiple index entries.[12]

Nevertheless, many people refer to such names as Western surnames, which purists might view as being incorrect. If you need to use the Western variant, you might have to use `\AccentCapThis\KempW` to get "À Kempis."

- Use `\PretagName` in all names with extended Unicode characters under `pdflatex`, NFSS, `inputenc`, and `makeindex`. See Section 2.8.4.
- `\CapThis\KempW` halts execution with `Argument of \UTFviii@two@ octets has an extra }`. Section 2.5.7 explains why.
- `\AccentCapThis\Name[Thomas]{à Kempis}` gives "Thomas ÀKempis" (space removed) in the absence of a non-breaking space or an explicit `\space.` This is due to TEX handling "`#1#2#3\relax`."
- Under `pdflatex` `\AccentCapThis` should not be used if the first character of a particle is not accented. `\AccentCapThis\Soto` gives DESoto. A particle like `lé` would cause `\AccentCapThis` to fail. Section 2.5.7 explains more.

To summarize, use non-breaking spaces when needed. With `pdflatex` and `inputenc`, use `\CapThis` when the first character of the particle is `a-z.` Use `\AccentCapThis` when the first character is extended Unicode.

[12] Properly speaking, "à Kempis" and "Aquinas" are not surnames but suffixed place names. They create different index entries from Western names and look different in the text.

Alternates

You could use name forms with braces, like `\Name[Thomas]{{à}~Kempis}`, and control sequences, like `\Name[Thomas]{\‘a~Kempis}`. Using those forms consistently, with `\PretagName,` would require you to use `\CapThis,` never `\AccentCapThis`. That is because both `{à}` and `\‘a` are passed as one macro argument, while under NFSS and `inputenc`, `à` is passed as two arguments in “`#1#2#3\relax`” (see Section 2.5.7).

Non-English contexts do not necessarily bind particles to surnames. Using `\Name` and `\FName` with alternate forenames helps address this and may skirt the particle capitalization issue. See also Section 2.10.2.

2.5.2 Formatting Initials

Omit spaces between initials if possible; see also Bringhurst’s *Elements of Typographic Style*. If your publisher wants spaces between initials, try putting thin spaces `\,` between them. Remember also to use `\retagName` to get the correct index sorting:

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

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

2.5.3 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}
```

We get 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}}
```

2.5.4 Listing by Surname

\ReverseCommaActive
\ReverseCommaInactive
\RevComma

The reversing macros `\ReverseCommaActive`, `\ReverseCommaInactive`, and `\RevComma` allow the easy generation of name lists ordered as ⟨*SNN*⟩, ⟨*FNN*⟩ or ⟨*SNN*⟩, ⟨*Alt. names*⟩. 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

Technical-Related Issues

2.5.5 Fault Tolerance

Especially since version 2.0, the nameauth package tries to prevent malformed input from creating side effects. For example, the malformed `\Name[Henry]{VIII}` no longer interferes with the well-formed `\Name{Henry}[VIII].` Furthermore, we guard against empty required values being passed to naming macros.

To reduce errors, `\Name`, `\FName`, `\AKA`, and `\IndexName` ignore leading and trailing spaces — but not medial spaces — 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.

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:

Macro Example	*periods*	*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..[13]

`\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 above.

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

2.5.7 Accented Names

For texts that contain accented characters, using `xelatex` or `lualatex` with `xindy` (`texindy`) is recommended. Since version 2.1, `nameauth` only requires special treatment of the leading character in the ⟨*SNN*⟩ field when you use `\CapThis` under `pdflatex`. In that case use `\AccentCapThis` (Section 2.5.1) if the character is not `a-z` or in the list below, for use with NFSS and the `utf8` input encoding:

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 ŀ 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

These characters really act as control sequences, so you must use `\PretagName` (Section 2.8.4) to sort them properly.

Examples `{æ}` and `\ae` always use argument `#1` if sent through the macro: `\def\foo#1#2#3\relax{<#1#2><#3>}`. With `\foo abc\relax` you always get <ab><c>. Both `\foo {æ}bc\relax` and `\foo \ae bc\relax` give <æb><c>.

Using `æ` by itself gets tricky. In `xelatex` and `lualatex` this character always uses `#1` in `\foo æbc\relax` to get <æb><c>. Using `pdflatex` with NFSS causes problems; `\foo æbc\relax` gives <æ><bc>. With `inputenc` and `fontenc` `æ` by itself uses `#1#2`. It is "two arguments wide." Any macro where this `#1#2` pair gets split into `#1` and `#2` will produce either the error `Unicode char ...not set up for LaTeX` or the error `Argument of \UTFviii@two@ octets has an extra }`. Using `\CapThis` can trigger this kind of error in `pdflatex`.

Unicode characters and control sequences are not interchangeable. The example below shows this difference because the names are all long instead of long, then short (if they were the same):

`\Name[Johann]{Andre\"a}`	JOHANN ANDREÄ
`\Name[Johann]{Andreä}`	JOHANN ANDREÄ instead of Andreä
`\Name{\AE thelred, II}`	ÆTHELRED II
`\Name{Æthelred, II}`	ÆTHELRED II instead of 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*). The following example lets you type, "In Congreſs, July 4, 1776."

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

Additionally, `\newunicodechar{ā}{\=a}` allows `\Name{Ghazāli}` to generate Ghazāli. But be careful! 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.

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
```

2.5.8 Custom Formatting

There are two kinds of formatting at work:

1. **Syntactic Formatting:** This includes reversing names, capitalizing the first letter in the ⟨*SNN*⟩ field in the body text, and capitalizing the root when ⟨*SNN*⟩ is a ⟨*root, suffix*⟩ pair.
2. **Typographic Formatting:** This happens after a name has been parsed and reordered as needed into the final form it will take in the text.

Continental small caps

Formatting does not affect the index. However, literal control sequences in the macro arguments of `\Name` and friends do make it into the index. Use this method with the `noformat` option to suppress default formatting. One also must use `\PretagName` to get proper index sorting:

```
\PretagName[Greta]{\textsc{Garbo}}{Garbo, Greta}
\Name[Greta]{\textsc{Garbo}}
```

You get Greta Garbo, then Garbo. Even if it "looks the same," the name `\Name[\normalfont{Greta}]{\textsc{Garbo}}` is a different name with a different index entry. In other words, avoid putting excess formatting into names, so that you gain both flexibility and consistency.

A comma delimiter will split the macro argument, potentially causing unbalanced braces. Avoid this by formatting the name and suffix separately:

```
\PretagName{\uppercase{Fukuyama}, Takeshi}{Fukuyama, Takeshi}
\PretagName[Thurston]{\textsc{Howell},\textsc{III}}%
  {Howell, Thurston 3}

\begin{nameauth}
  \< Fukuyama & & \uppercase{Fukuyama}, Takeshi & >
  \< Howell & Thurston & \textsc{Howell},\textsc{III} & >
\end{nameauth}
```

`\Fukuyama` produces FUKUYAMA Takeshi and FUKUYAMA. Of course, you could type all-capital surnames without control sequences. Likewise, `\Howell` generates Thurston Howell III and Howell.

`\NameauthName` `\NameauthLName` `\NameauthFName`

These macros are set by default to `\@nameauth@Name`, the internal name parser. The main and simplified interfaces call them as respective synonyms for `\Name, \Name*,` and `\FName.` Should you desire to create your own naming macros, you can redefine them. Here is the minimal working example:

```
\makeatletter
\newcommandx*\MyName[3][1=\@empty, 3=\@empty]%
  {⟨Name here⟩}%
\newcommandx*\MyLName[3][1=\@empty, 3=\@empty]%
  {⟨Long name here⟩\@nameauth@FullNamefalse}%
\newcommandx*\MyFName[3][1=\@empty, 3=\@empty]%
  {⟨Short name here⟩\@nameauth@FirstNamefalse}%
\makeatother
```

The macros above do not really work together with the rest of nameauth package, so be careful! You can hook these macros into the user interface thus:

```
\renewcommand*{\NameauthName}{\MyName}
\renewcommand*{\NameauthLName}{\MyLName}
\renewcommand*{\NameauthFName}{\MyFName}
\begin{nameauth}
  \< Silly & No Particular & Name & >
\end{nameauth}
This is \Silly, \LSilly, and \SSilly.
```

This is ⟨*Name here*⟩, ⟨*Long name here*⟩, and ⟨*Short name here*⟩.

Like `\NamesFormat` below, the macros `\NameauthName, \NameauthLName,` and `\NameauthFName` respect scoping unless you use `\global.` Now `\Silly` and `\Name[No Particular]{Name}` produce No Particular Name and Name.

`\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 formatting in all footnotes:

```
\makeatletter
\let\@oldfntext\@makefntext
\long\def\@makefntext#1{%
  \renewcommand*\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 local scope of `\@makefntext`.

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.

Outside of the `quote` environment above, we have Vlad III Dracula and Vlad III. For references to "Vlad" instead of "Vlad III" one could use `\Name{Vlad, III Dracula}`. Do not mix these forms with each other or with the old syntax, lest errors bite! You would get multiple index entries, unwanted cross-references, and unexpected forms in the text. The simplified interface helps one to avoid this.

2.5.9 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 in footnotes to implement a different system of names (see also Section 2.6.2):

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

2.6 Name Decisions

2.6.1 Testing Decisions

Sometimes the behavior of your document may need to change if a name is present or not. The macros in this subsection help to make such changes possible. Only `\Name`, `\Name*`, `\PName`, `\AKA`, `\AKA*`, `\ForgetName`, `\SubvertName`, and `\ExcludeName` affect the results of these macros.

The following macros can be useful for generating conditional output in the document, such as inserting a "short biography" footnote or callout text after a name, depending on a particular context. Authors also can use them to generate "reminders to self" if a name has or has not occurred yet in the document. Such reminders could be linked with the comment, pdfcomment, and similar packages to aid writing and name management.

\IfMainName

If you want to produce output or perform a task based on whether a "main body" name exists, use `\IfMainName`, whose syntax is:

`\IfMainName[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[`⟨*Alternate names*⟩`]{`⟨*yes*⟩`}{`⟨*no*⟩`}`

This is a long macro via `\newcommandx`, so you can have paragraph breaks in the ⟨*yes*⟩ and ⟨*no*⟩ paths. A "main body" name is capable of being formatted by this package, *i.e.*, one created by the naming macros when the `mainmatter` option is used or after `\NamesActive`. It is distinguished from those names that occur in the front matter and those that have been used as cross-references.

For example, we can test if we have yet to encounter the name "Bob Hope" in this document. We get "I have not met Bob Hope" from the following:

```
\IfMainName[Bob]{Hope}%
{I met \Name[Bob]{Hope}}%
{I have not met \Name[Bob]{Hope}}
```

We will meet Bob in Section 2.7.1, so after this example we need to use the macro `\ForgetName[Bob]{Hope}` to "forget" that we saw this name.

`\IfFrontName` If you want to produce output or perform a task based on whether a "front matter" name exists, use `\IfFrontName`, whose syntax is:

`\IfFrontName[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[`⟨*Alternate names*⟩`]{`⟨*yes*⟩`}{`⟨*no*⟩`}`

This macro works the same as `\IfMainName`. A "front matter" name is not capable of being formatted by this package, *i.e.*, one created by the naming macros when the `frontmatter` option is used or after `\NamesInactive`. It is distinguished from those names that occur in the main matter and those that have been used as cross-references.

For example, we can test whether a name has been used as a formatted name, an unformatted name, or both:

```
\IfFrontName[Rudolph]{Carnap}%
{\IfMainName[Rudolph]{Carnap}%
  {\Name[Rudolph]{Carnap} is both}%
  {\Name[Rudolph]{Carnap} is only non-formatted}}%
{\IfMainName[Rudolph]{Carnap}%
  {\Name[Rudolph]{Carnap} is only formatted}%
  {\Name[Rudolph]{Carnap} is not mentioned}}
```

From this we get the result that we expect from Section 2.5.9, namely: Carnap is both. We will return to this topic later.

`\IfAKA` If you want to produce output or perform a task based on whether a "*see*-reference" name exists, use `\IfAKA`, whose syntax is:

`\IfAKA[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[`⟨*Alt. names*⟩`]{`⟨*y*⟩`}{`⟨*n*⟩`}{`⟨*excluded*⟩`}`

This macro works similarly to `\IfMainName`, although it has an "excluded" branch in order to detect those names excluded from indexing by `\ExcludeName` (Section 2.8.7). A "*see*-reference" name is printed in the body text but only exists as a cross-reference created by `\AKA` and `\AKA*`. From the following example we get "John David Rockefeller IV has an alias":

```
\IfAKA[Jay]{Rockefeller}%
  {\LJRIV\ has an alias}%
  {\LJRIV\ does not have an alias}%
  {\LJRIV\ is excluded}
```

If you are confident that you will not be dealing with names generated by `\ExcludeName` then you can just leave the ⟨*excluded*⟩ branch as `{}`.

A similar use of `\IfAKA{Confucius}` tells us that "Confucius is not an alias." Yet one might have to check further about Confucius to see if it is used as a formatted name or a non-formatted name.

Here we test for a name used with `\ExcludeName` (Section 2.8.7) to get the result, "GRINCH is excluded":

```
\ExcludeName{Grinch}%
\IfAKA{Grinch}%
  {\Name{Grinch} is an alias}%
  {\Name{Grinch} is not an alias}%
  {\Name{Grinch} is excluded}
```

2.6.2 Changing Decisions

This section describes macros that change the status of whether a name has occurred. That also helps to avoid clashes between formatted and non-formatted names. They are meant for editing at or near the final draft stage. "*See*-reference" names created by `\AKA` are not affected by these macros.

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

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

This macro causes `\Name` and friends globally 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.

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

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

This macro causes `\Name` and friends globally 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.

Scope The default behavior of these two macros changes whether a name is "forgotten" or "subverted" simultaneously for front matter and main matter names, Remember the example above that gave us the answer, "Carnap is both":

```
\IfFrontName[Rudolph]{Carnap}%
{\IfMainName[Rudolph]{Carnap}%
  {\Name[Rudolph]{Carnap} is both}%
  {\Name[Rudolph]{Carnap} is only non-formatted}}%
{\IfMainName[Rudolph]{Carnap}%
  {\Name[Rudolph]{Carnap} is only formatted}%
  {\Name[Rudolph]{Carnap} is not mentioned}}
```

Now watch closely: After we use `\ForgetName[Rudolph]{Carnap}` we get the result, "Rudolph Carnap is not mentioned." Both the main matter name and the front matter name were forgotten!

This default behavior helps synchronize formatted and unformatted types of names. For example, if you wanted to use unformatted names in the footnotes and formatted names in the text (Section 2.5.9), you could use, *e.g.* `\SubvertName` right after the first use of a name in the body text, ensuring that all references in the text and notes would be short unless otherwise modified.[14]

`\LocalNames` `\GlobalNames`

If, however, this "global" behavior of `\ForgetName` and `\SubvertName` is not desired, you can use `\LocalNames` to change that behavior and `\GlobalNames` to restore the default behavior. Both of these macros are global in scope.

After `\LocalNames,` if you are in a "front matter" section via the `frontmatter` option or `\NamesInactive, \ForgetName` and `\SubvertName` will only affect unformatted names. If you are in a "main matter" section via the `mainmatter` option or `\NamesActive,` then `\ForgetName` and `\SubvertName` will only affect formatted names. The following example helps to illustrate local changes:

```
\def\CheckChuck{%\IfFrontName[Charlie]{Chaplin}%
  {\IfMainName[Charlie]{Chaplin}{both}{front}}%
  {\IfMainName[Charlie]{Chaplin}{main}{none}}}%
```

We begin by creating a formatted name in the main matter:

```
\Name*[Charlie]{Chaplin}
\CheckChuck
```

Charlie Chaplin
main

Now we switch to an unformatted section and create a name there:

```
\NamesInactive
\Name*[Charlie]{Chaplin}
\CheckChuck
```

Charlie Chaplin
both

We now have two names. They look and behave the same, but are two different "species" with independent first and subsequent uses. We set the scope of `\ForgetName` and `\SubvertName` to local, then forget the name in the unformatted section:

```
\LocalNames
\ForgetName[Charlie]{Chaplin}
\CheckChuck
```

main

We "subvert" the name control sequence in the unformatted section, as if the name has occurred, and switch back to the main section:

```
\SubvertName[Charlie]{Chaplin}
\NamesActive
\CheckChuck
```

both

We forget the formatted name and reset the default behavior:

```
\ForgetName[Charlie]{Chaplin}
\GlobalNames
\CheckChuck
```

front

Finally, we forget everything:

```
\ForgetName[Charlie]{Chaplin}
\CheckChuck
```

none

[14] This manual takes advantage of that behavior at times in order to synchronize first and subsequent uses of names between formatted and unformatted sections of the body text.

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. 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, similar to the naming macros.

\AKA* has two functions. For Western names, where ⟨*Alt. FNN*⟩ is present, \AKA* prints either just the ⟨*Alt. FNN*⟩ or just the ⟨*Alt. names*⟩ when they also are present. However, if ⟨*Alt. FNN*⟩ is absent, \AKA* prints just ⟨*Alt. names*⟩ if present, otherwise ⟨*Alt. SNN*⟩. See also Section 2.8.5.

Here is a simple example with the default system of formatting:

```
\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.

"Jean the Fearless" intentionally receives no formatting because I intend that only names with main index entries be formatted. One could put \AKA within a formatting macro to make that otherwise. Still, the reversing and capitalizing mechanisms do work with \AKA.

This complex example shows the Continental style with no default formatting:

```
\PretagName[Heinz]{\textsc{Rühmann}}{Ruehmann, Heinz}%
\PretagName[Heinrich Wilhelm]{\textsc{Rühmann}}%
  {Ruehmann, Heinrich Wilhelm}%
\SubvertName[Heinz]{\textsc{Rühmann}}%
\AKA*[Heinz]{\textsc{Rühmann}}%
  [Heinrich Wilhelm]{\textsc{Rühmann}} %
``\FName[Heinz]{\textsc{Rühmann}}'' %
\Name[Heinz]{\textsc{Rühmann}} %
(7 March 1902--3 October 1994) was a German film actor who
appeared in over 100 films between 1926 and 1993.
```

Heinrich Wilhelm "Heinz" Rühmann (7 March 1902–3 October 1994) was a German film actor who appeared in over 100 films between 1926 and 1993.

First I want "Heinz Rühmann" to be the main name of reference, so \SubvertName causes \FName to print the short version. Second, I use \AKA* to print only the forenames "Heinrich Wilhelm" in the body text. Nevertheless, the index cross-reference will be complete. Third, \FName prints "Heinz." Finally, \Name prints Rühmann. Of course I used \PretagName to sort the index entries.

Using Bob Hope, Louis XIV, Lao-tzu, and Gregory I as examples, we see how `\AKA` and `\AKA*` work:

`\AKA[Bob]{Hope}[Leslie Townes]{Hope}`	Leslie Townes Hope
`\AKA*[Bob]{Hope}[Leslie Townes]{Hope}`	Leslie Townes
`\AKA[Bob]{Hope}%` `  [Leslie Townes]{Hope}[Lester T.]`	Lester T. Hope
`\AKA*[Bob]{Hope}%` `  [Leslie Townes]{Hope}[Lester T.]`	Lester T.
`\AKA{Louis, XIV}{Sun King}`	Sun King
`\AKA*{Louis, XIV}{Sun King}`	Sun King
`\AKA{Lao-tzu}{Li, Er}`	Li Er
`\AKA*{Lao-tzu}{Li, Er}`	Li Er
`\AKA{Gregory, I}{Gregory}[the Great]`	Gregory the Great
`\AKA*{Gregory, I}{Gregory}[the Great]`	the Great

The alternate form "Lester T. Hope" does not appear in the index, but only in the body text to allow for variations. Gregory I "the Great" happens with

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

The index has a *see* reference from "Gregory the Great" to "Gregory I." Some helpful tips for `\AKA` and `\AKA*` follow:

- `[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}` is the main name. `[`⟨*Alt. FNN*⟩`]{`⟨*Alt. SNN*⟩`}[`⟨*Alt. names*⟩`]` is the cross-reference. Forgetting this may cause errors.
- The old syntax causes `\AKA` and `\AKA*` to fail: `\AKA{Louis}[XIV]{Sun King}` and `\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 as pseudonyms, with all that entails. 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
  `\RevName`: Yakumo KOIZUMI
- The name fields of `\PretagName` correspond with the `[`⟨*Alt. FNN*⟩`]{`⟨*Alt. SNN*⟩`}[`⟨*Alt. names*⟩`]` fields of `\AKA`:

  `\AKA{Vlad III, Dracula}{Vlad, Ţepeş}` matches
  `\PretagName{Vlad, Ţepeş}{Vlad Tepes}`

  This form does not match: `\PretagName{Vlad}[Ţepeş]{Vlad Tepes}`.
- With stage names like The Amazing Kreskin, if you want them in the index, use `\Name[The Amazing]{Kreskin}` to get "Kreskin, The Amazing." Otherwise use something like `\Name[J.]{Kreskin}[The Amazing]` to get The Amazing Kreskin in the text and "Kreskin, J." in the index.

  Using `\AKA` with such names looks like: `\AKA[The Amazing]{Kreskin}[Joseph]{Kresge}` and `\AKA[J.]{Kreskin}[Joseph]{Kresge}`.You get The Amazing Kreskin, a.k.a. Joseph Kresge.

- 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. This may be the easiest solution.
  2. 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. This is the best solution in terms of how nameauth is designed.
  3. Always get “Iron Mike Tyson” with something like:

     ```
     \newcommand*{\Iron}{\SubvertName[Mike]{Tyson}%
     \FName[Mike]{Tyson}[Iron Mike] \Name[Mike]{Tyson}}
     ```

     ``\Iron'' gives you “Iron Mike Tyson.”[15] You are responsible for typesetting the first use and creating a cross-reference. This solution runs somewhat contrary to the design principles of nameauth, but it may be helpful if you want the invariant name “Iron Mike Tyson” to recur and you want to save typing.

`\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 2.11. 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.[16]

[15] In typesetting this manual I defined the macro `\Iron` and others like it on one continuous line because defining a macro over multiple lines with comment characters ending them in ltxdoc and a `.dtx` file caused extra spaces to be inserted.

[16] 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.

Using \PretagName helps avoid the need for manual index entries. Instead of doing a lot of extra work for some names, consider the following example:

```
\PretagName{\textit{Doctor Angelicus}}%
  {Doctor Angelicus}%
Perhaps the greatest medieval theologian was
\Name{Thomas, Aquinas}
(\AKA{Thomas, Aquinas}{Thomas of Aquino}),
also known as
\AKA{Thomas, Aquinas}{\textit{Doctor Angelicus}}.
```

Perhaps the greatest medieval theologian was Thomas Aquinas (Thomas of Aquino), also known as *Doctor Angelicus*.

We use the medieval form: \Name{Thomas, Aquinas} because "Aquinas" is not a surname. Section 2.5.1 talks about those unfortunate situations where one must use the Western form \Name[Thomas]{Aquinas}.

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. Neither \AKA*, nor \CapName, \CapThis, \RevComma, \RevName, and the related package options work with \PName.

The main name comes first, followed by the name that is only a *see* reference. \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)

\PName can be a bit sketchy with medieval names. You get William I (William the Conqueror) with \PName{William, I}{William, the Conqueror}. Stay away from \PName{William, I}{William}[the Conqueror] because that is the old syntax that can break both \AKA and \PName if used in the leading arguments instead of in the trailing arguments. The old syntax can get you confused and lead you to type \PName{William, I}[William]{the Conqueror}. You would get a name that looked right in the body text but wrong in the index.

Something like \PName{Lao-tzu}{Li, Er} "Lao-tzu (Li Er)" works well enough, but \PName{Gregory, I}{Gregory}[the Great] "Gregory I (Gregory the Great)" starts moving close to the old syntax.

2.8 Indexing Macros

2.8.1 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.

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

2.8.2 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` redefines `\textlatin` to produce the page number itself within a certain scope like:

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

This solution has proven quite effective for me. One need only type

```
\fixindex{%
⟨paragraphs of running text⟩%
}
```

Of course, one can opt to check if `\textlatin` is defined, save its value, redefine it, then restore it, perhaps even in an environment.

2.8.3 `\IndexName`

\IndexName

This macro creates an index entry like those created by `\Name` and friends. It prints nothing in the body text. The syntax is:

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

`\IndexName` complies with the new syntax, where a suffixed pair in ⟨*SNN*⟩ is a name/affix pair that can be ancient or Eastern. If ⟨*FNN*⟩ are present, it ignores ⟨*Alternate names*⟩. Otherwise, if ⟨*FNN*⟩ are absent, `\IndexName` sees ⟨*Alternate names*⟩ as an affix using the old syntax.

After `\IndexInactive` this macro does nothing until `\IndexActive` appears. 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. Otherwise, if ⟨*FNN*⟩ is absent, the comma would trigger ancient, medieval, and Eastern name forms in the index.

2.8.4 Index Sorting

The general practice for sorting with `makeindex -s` 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 names like Jan Łukasiewicz and Æthelred II by putting them between `\IndexInactive` and `\IndexActive` while creating manual index entries.

`\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 differs from the other tagging macros:

- 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."

`\PretagName` creates a sort key terminated with the "actual" character, which is `@` by default. Do not include the "actual" character in the pretag. Here is an example of its use:

```
\PretagName[Jan]{Łukasiewicz}{Lukasiewicz, Jan}
\PretagName{Æthelred, II}{Aethelred 2}
```

One need only pretag names once in the preamble. Every time that one refers to Łukasiewicz or Æthelred, 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.

2.8.5 `\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 are not "pretags." To help sort that out, we look at what gets affected by these commands:

	`\PretagName`		
`\index{`	`Aethelred 2@`	`Æthelred II`	`, king}`
			`\TagName` and `\UntagName`

All the tagging commands use the name arguments as a reference point. `\PretagName` generates the leading sort key while `\TagName` and `\UntagName` affect the trailing content of the index entry.

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}                    (in the preamble)
\TagName{Gregory, I}{, pope}
...
\Name*{Leo, I} \Name*{Gregory, I}           (first references to Leo I
                                             and Gregory I)
...
\Name*{Leo, I} was known as                 Leo I was known as Leo
\AKA{Leo, I}{Leo}[the Great].               the Great.
...
\Name{Gregory, I} ``\AKA*{Gregory, I}%      Gregory "the Great," an-
{Gregory}[the Great],'' another major       other major pope.
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.

2.8.6 \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.[17]

2.8.7 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*⟩`]`

You will see excluded names printed in the body text with all the formatting and other features:

```
\ExcludeName[Kris]{Kringle}
\Name[Kris]{Kringle} and \Name[Kris]{Kringle}:
```
Kris Kringle and Kringle.

Nevertheless, no matter how many times you use Kringle in the body text, the name will never appear in the index. Remember the Grinch from Section 2.6.1? He will not appear in the index either.

`\ExcludeName` also prevents cross-references. You may see output in the body text, but no *see*-reference will appear in the index:

```
\ExcludeName[Santa]{Claus}
\AKA[Kris]{Kringle}[Santa]{Claus} Santa Claus
```

Instead of using `\ExcludeName`, which basically prevents the indexing mechanism of the naming macros from doing anything with a particular name, it is far likelier that you would use the index control macros (Section 2.8.1).

[17] 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.

2.9 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.

2.11 Errors and Warnings

Here are some ways to avoid common errors:

- 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."
- Do not apply a formatting macro to an entire comma-delimited ⟨*SNN, affix*⟩ pair. `\Name[Oskar]{\textsc{Hammerstein, II}}` fails due to unbalanced braces because it gets split up. Format each part instead *e.g.*, `\Name[Oskar]{\textsc{Hammerstein},\textsc{II}}`.
- With `pdflatex` use `\CapThis` when the first letter of a surname particle is `a-z`, otherwise use `\AccentCapThis` if it is extended Unicode. Doing otherwise may cause unbalanced braces and related errors.
- Consider using `\PretagName` with all names containing control sequences or extended Unicode; see Section 2.8.4.
- 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 older syntax presents its own group of potential errors:

- Erroneously typing `\Name[Henry]{VIII}` prints "Henry VIII" and "VIII," as well as producing a malformed index entry.
- Avoid forms like `\Name[Henry]{VIII}[Tudor]` which gives "Tudor VIII" and "VIII." This is a Western name form, not an ancient form. It may act as malformed input if you mix it with proper medieval name forms, but it will not affect them adversely.
- 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."

Warnings result from the following:

- Using a cross-reference `[`⟨*Alternate names*⟩`]{`⟨*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:

  ```
  \PretagName[E.\,B.]{White}{White, E. B.}...
  ```

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

  Such redefinitions could generate unwanted index entries.

3 Implementation

3.1 Boolean Values

Affix Commas

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
```

Toggle Formatting

`\NamesActive` and `\NamesInactive` or the `mainmatter` and `frontmatter` options set this value below.

```
\newif\if@nameauth@DoFormat
```

The next value works with `\LocalNames` and `\GlobalNames`.

```
\newif\if@nameauth@LocalNames
```

Indexing

`\IndexActive` and `\IndexInctive` or the `index` and `noindex` options set this below:

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

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

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

Name Formatting

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
```

`@nameauth@FirstFormat` toggles the formatting of first occurrences of names.
`@nameauth@AlwaysFormat` forces name formatting whenever formatting is active.

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

`@nameauth@FullName` toggles long or short forms in subsequent name uses.
`@nameauth@FirstName` is used when printing only first names. `@nameauth@AltAKA` is toggled by either `\AKA` or `\AKA*` to print a longer or shorter name.

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

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

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

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

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

This Boolean value is triggered by `\CapThis`. Each instance of `\Name` and `\AKA` reset it.

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

This Boolean value is triggered by `\AccentCapThis` to handle special cases of extended Unicode particle caps. Each instance of `\Name` and `\AKA` reset it.

```
\newif\if@nameauth@Accent
```

3.2 Hooks

`\NamesFormat` Change typographic formatting of final complete name form in text. See Section 2.5.8.

```
\newcommand*{\NamesFormat}{}
```

`\NameauthName` Hook to create custom naming macros. Usually these three macros have the same control sequence, but they need not do so if you want something different. See Section 2.5.8.

```
\newcommand*{\NameauthName}{\@nameauth@Name}
```

`\NameauthLName` Hook to create custom naming macros. Called after `@nameauth@FullName` is set true. See Section 2.5.8.

```
\newcommand*{\NameauthLName}{\@nameauth@Name}
```

`\NameauthFName` Hook to create custom naming macros. Called after `@nameauth@FirstName` is set true. See Section 2.5.8.

```
\newcommand*{\NameauthFName}{\@nameauth@Name}
```

3.3 Package Options

The following package options interact with many of the prior Boolean values.

```
\DeclareOption{comma}{\@nameauth@AlwaysCommatrue}
\DeclareOption{nocomma}{\@nameauth@AlwaysCommafalse}
\DeclareOption{mainmatter}{\@nameauth@DoFormattrue}
\DeclareOption{frontmatter}{\@nameauth@DoFormatfalse}
\DeclareOption{index}{\@nameauth@DoIndextrue}
\DeclareOption{noindex}{\@nameauth@DoIndexfalse}
\DeclareOption{pretag}{\@nameauth@Pretagtrue}
\DeclareOption{nopretag}{\@nameauth@Pretagfalse}
\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{alwaysformat}{\@nameauth@AlwaysFormattrue}
\DeclareOption{smallcaps}{\renewcommand*{\NamesFormat}{\scshape}}
\DeclareOption{italic}{\renewcommand*{\NamesFormat}{\itshape}}
\DeclareOption{boldface}{\renewcommand*{\NamesFormat}{\bfseries}}
\DeclareOption{noformat}{\renewcommand*{\NamesFormat}{}}
\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{ifluatex}
\RequirePackage{ifxetex}
\RequirePackage{trimspaces}
\RequirePackage{suffix}
\RequirePackage{xargs}
```

This macro determines the "actual" character for index sorting.

```
\newcommand*\@nameauth@Actual{@}
```

3.4 Internal Macros

`\@nameauth@Clean` Thanks to Heiko Oberdiek, this macro produces a "sanitized" string, even using accented characters, based on the arguments 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 arguments) 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. In `xelatex` and `lualatex` this is trivial and causes no problems. In `pdflatex` we have to account for "double-wide" accented Unicode characters.

```
\newcommand*{\@nameauth@CapRoot}[1]%
{%
  \ifxetex
    \@nameauth@CRii#1\relax%
  \else
    \ifluatex
      \@nameauth@CRii#1\relax%
    \else
      \if@nameauth@Accent
        \@nameauth@CRiii#1\relax%
      \else
        \@nameauth@CRii#1\relax%
      \fi
    \fi
  \fi
}
```

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

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

`\@nameauth@CRiii` This is called in `pdflatex` under `inputenc` where an accented Unicode character takes the first two arguments. Grab the first "letter" as two arguments and cap it, then everything before `\relax` as the third. Capitalize the first and return it with the second.

```
\def\@nameauth@CRiii#1#2#3\relax{\uppercase{#1#2}\@nameauth@Root{#3}}
```

`\@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 argument. 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 argument. 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 2.5.8) 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
}
```

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

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

Names occur in horizontal mode; we ensure that. Next we make copies of the arguments to test them and make parsing decisions

```
  \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}}%
```

Test for malformed input.

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

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

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

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}}%
\if@nameauth@DoCaps
  \protected@edef\CapShort{\@nameauth@CapRoot{#2}}%
\else
  \let\CapShort\RawShort%
\fi
\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
  \else
```

This is the section that handles Western names and non-native Eastern names. The first pair of conditionals handle the `comma` option, `\RevThisComma`, 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@Accentfalse%
  \@nameauth@AllThisfalse%
  \@nameauth@ShowCommafalse%
  \@nameauth@RevThisfalse%
  \@nameauth@RevThisCommafalse%
```

Call the full stop detection.

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

3.5 User Interface Macros

`\CapThis` Tells the root capping macro to cap an unaccented character.

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

`\AccentCapThis` Tells the root capping macro to cap an accented Unicode character.

```
\newcommand*{\AccentCapThis}{\@nameauth@Accenttrue\@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 Change the "actual" character from the default.

```
\newcommand*{\IndexActual}[1]%
  {\global\renewcommand*\@nameauth@Actual{#1}}
```

\LocalNames \LocalNames sets @nameauth@LocalNames true so \ForgetName and \SubvertName do not affect both formatted and unformatted names.

```
\newcommand*\LocalNames{\@nameauth@LocalNamestrue}
```

\GlobalNames \GlobalNames sets @nameauth@LocalNames false, restoring the default behavior of \ForgetName and \SubvertName.

```
\newcommand*\GlobalNames{\@nameauth@LocalNamesfalse}
```

\Name \Name calls \NameauthName, the interface hook.

```
\def\Name{\NameauthName}
```

\Name* \Name* sets up a long name reference and calls \NameauthLName, the interface hook.

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

\FName \FName sets up a short name reference and calls \NameauthFName, the interface hook.

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

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

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

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

```
  \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}}%
```

Test for malformed input.

```
  \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
```

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

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

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}}%
\if@nameauth@DoCaps
  \protected@edef\CapShort{\@nameauth@CapRoot{#4}}%
\else
  \let\CapShort\Shortd
\fi
\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@AltAKA
      \ex\@nameauth@FmtName\ex{\FNN}%
    \else
      \if@nameauth@RevThis
        \ex\@nameauth@FmtName\ex{\ex\SNNd\ex\Space\FNN}%
      \else
        \ex\@nameauth@FmtName\ex{\ex\FNN\ex\space\SNNd}%
      \fi
    \fi
  \fi
```

Reset all the "per name" Boolean values.

```
  \@nameauth@NBSPfalse%
  \@nameauth@AltAKAfalse%
  \@nameauth@DoCapsfalse%
  \@nameauth@Accentfalse%
  \@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 argument, if that exists, and calls `\AKA`.

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

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

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

\PName* This sets up a long name reference and calls `\PName`.

```
\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
}
```

`\IfAKA` This macro expands one path if a see-reference name exists, another if it does not exist, and a third if it is excluded.

```
\newcommandx\IfAKA[6][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
        \ifcsempty{\csb!PN}{#4}{#6}%
      \else#5\fi
    \else
      \ifcsname\csbc!PN\endcsname
        \ifcsempty{\csbc!PN}{#4}{#6}%
      \else#5\fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \ifcsempty{\csab!PN}{#4}{#6}%
    \else#5\fi
  \fi
}
```

`\IfFrontName` This macro expands one path if a front matter name exists, or else the other if it does not exist.

```
\newcommandx\IfFrontName[5][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!NF\endcsname#4\else#5\fi
    \else
      \ifcsname\csbc!NF\endcsname#4\else#5\fi
    \fi
  \else
    \ifcsname\csab!NF\endcsname#4\else#5\fi
  \fi
}
```

`\IfMainName` This macro expands one path if a main matter name exists, or else the other if it does not exist.

```
\newcommandx\IfMainName[5][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!MN\endcsname#4\else#5\fi
    \else
      \ifcsname\csbc!MN\endcsname#4\else#5\fi
    \fi
  \else
    \ifcsname\csab!MN\endcsname#4\else#5\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.

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

Now we parse the arguments, undefining the control sequences either locally by section type or globally. `@nameauth@LocalNames` toggles the local or global behavior, while `@nameauth@DoFormat` selects the type of name.

```
  \ifx\testa\@empty
    \ifx\testc\@empty
      \if@nameauth@LocalNames
        \if@nameauth@DoFormat
          \global\csundef{\csb!MN}%
        \else
          \global\csundef{\csb!NF}%
        \fi
      \else
        \global\csundef{\csb!MN}%
        \global\csundef{\csb!NF}%
      \fi
    \else
      \if@nameauth@LocalNames
        \if@nameauth@DoFormat
          \global\csundef{\csbc!MN}%
        \else
          \global\csundef{\csbc!NF}%
        \fi
      \else
        \global\csundef{\csbc!MN}%
        \global\csundef{\csbc!NF}%
      \fi
    \fi
  \else
    \if@nameauth@LocalNames
      \if@nameauth@DoFormat
        \global\csundef{\csab!MN}%
      \else
        \global\csundef{\csab!NF}%
      \fi
```

```
    \else
      \global\csundef{\csab!MN}%
      \global\csundef{\csab!NF}%
    \fi
  \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.

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

Now we parse the arguments, defining the control sequences either locally by section type or globally. @nameauth@LocalNames toggles the local or global behavior, while @nameauth@DoFormat selects the type of name.

```
  \ifx\testa\@empty
    \ifx\testc\@empty
      \if@nameauth@LocalNames
        \if@nameauth@DoFormat
          \csgdef{\csb!MN}{}%
        \else
          \csgdef{\csb!NF}{}%
        \fi
      \else
        \csgdef{\csb!MN}{}%
        \csgdef{\csb!NF}{}%
      \fi
    \else
      \if@nameauth@LocalNames
        \if@nameauth@DoFormat
          \csgdef{\csbc!MN}{}%
        \else
          \csgdef{\csbc!NF}{}%
        \fi
      \else
        \csgdef{\csbc!MN}{}%
        \csgdef{\csbc!NF}{}%
      \fi
    \fi
```

```
  \else
    \if@nameauth@LocalNames
      \if@nameauth@DoFormat
        \csgdef{\csab!MN}{}%
      \else
        \csgdef{\csab!NF}{}%
      \fi
    \else
      \csgdef{\csab!MN}{}%
      \csgdef{\csab!NF}{}%
    \fi
  \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\csgdef\ex{\ex\arga\ex}\ex{\ex\NameauthName\ex{\the\tokc}}%
            \ex\csgdef\ex{\ex L\ex\arga\ex}\ex{\ex\@nameauth@FullNametrue%
              \ex\NameauthLName\ex{\the\tokc}}%
            \ex\csgdef\ex{\ex S\ex\arga\ex}\ex{\ex\@nameauth@FirstNametrue%
              \ex\NameauthFName\ex{\the\tokc}}%
```

```
          \else
            \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\arga\ex\ex\ex}%
              \ex\ex\ex{\ex\ex\ex\NameauthName\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\@nameauth@FullNametrue%
              \ex\ex\ex\NameauthLName\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\@nameauth@FirstNametrue%
              \ex\ex\ex\NameauthFName\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\NameauthName\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\@nameauth@FullNametrue%
              \ex\ex\ex\NameauthLName\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\@nameauth@FirstNametrue%
              \ex\ex\ex\NameauthFName\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\NameauthName%
              \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\@nameauth@FullNametrue%
              \ex\ex\ex\ex\ex\ex\ex\NameauthLName%
              \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\@nameauth@FirstNametrue%
              \ex\ex\ex\ex\ex\ex\ex\NameauthFName\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.