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

Charles P. Schaum†

Released 2016/09/19

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 file describes version v2.6, last revised 2016/09/19.

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

1 Introduction

1.1 Preliminary Information

Disclaimer

This manual uses names of living and dead 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 herein are used only for teaching purposes. Especially in the index, fictional names have an asterisk (*), Eastern names with Western index forms have a dagger (†), and names that use the older syntax have a double dagger (‡).

Design

When publications use hundreds of names, it takes time and money to check them. This package automates much of that work:

- **Automation** of name forms in the text aids professional writing and allows the arbitrary replacement of text without retyping names.
  - 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.
- Name variants in the text still produce **consistent index entries**.
- One can design **complex name formatting.**
- One can **automate information retrieval** about names.
- One can implement and automate a **name authority,** a master list of names that permits known name variants.
- Some **cross-cultural naming conventions** are possible.
- **Automatic sort keys and tags** aid indexing.
- Provide **automatic name presentation** that could work in a LaTeX backend for a document transform, database report, etc.

This package grew from generalized needs for translating old German and Latin texts into English. One can enable Continental and non-English standards via control sequences in the macro arguments or through bypassing and reformatting the post-processing modules (Sections 2.5.5, 2.5.6, 2.9.4, and 2.10.8).

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.

Technical Notes

- We take an "easy stuff first" approach and indicate when the number or complexity of concepts might be confusing.

- When this manual touches on some of the more esoteric points of this package that require more caution, it uses the "dangerous bend."
- This manual uses package macros and internals in special ways to isolate and show its examples. In some cases we point that out, while in others we assume that the reader understands this.

- With version 2.5 no formatting is selected by default. To implement formatting, see Sections 2.5.6, 2.5.10, and 2.10.6ff. These sections can be more complex than others.
- Version 2.6 fixes several issues with the `comma` option and with the old syntax. The `comma` option no longer affects Eastern and ancient names. The old and new syntax are interchangeable, save with `\AKA` and its derivatives. See Sections 2.2.3, 2.4.1, and 2.8.1.
- This package really should be rewritten at some point. The author offers his humble apologies for any current shortcomings.[1]
- This manual is designed to be compatible with both A4 and US letter stock size formats.
- Macro references are minimized for a "clean" index, showing how nameauth "normally" handles indexing.
- The nameauth package requires etoolbox, ifxetex, ifluatex, suffix, trimspaces, and xargs.
- With each release, this documentation is tested with dvi-mode `latex`, `pdflatex`, `lualatex`, and `xelatex` using `makeindex`. This is done by running the GNU Makefile with the ENGINE=⟨*engine*⟩ option.
- Only the `pdflatex` version is released with the package.
- The package also works with `xindy`, although `\PretagName` may not be necessary in that case. Using `xindy` is best for languages that have sort orders for letters that cannot map to English. German *does* map to English: Ä, Ö, Ü, and ß are AE, OE, UE, and SS. Norwegian *does not* map to English: Æ, Ø, and Å come after Z in that order.
- This manual was typeset with `pdflatex` using `makeindex` and `gind.ist`. (This item changes, depending on the LaTeX engine).

Thanks

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. Thanks also to the users of this package, whose feedback has helped me do a better job than I could have done otherwise.

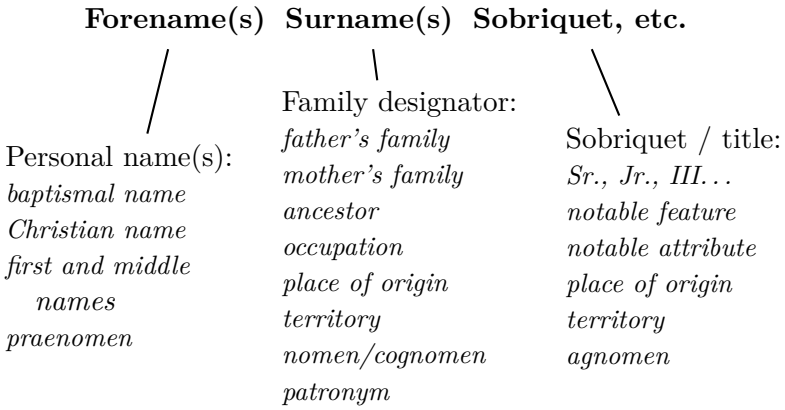
[1] Future possibilities include: `\RevName` and friends force long name references; `\RevComma` and friends work only on Western names, independent of the present reversing system; a better way of parsing names and setting values will be implemented.

1.2 What's In A Name?

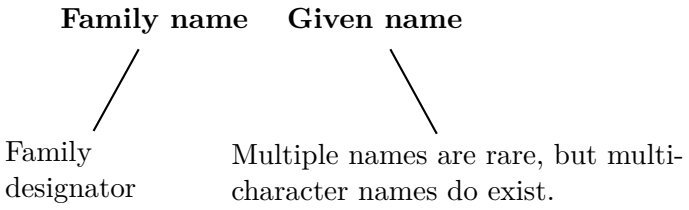
Name forms are ambiguous apart from historical and cultural contexts. The nameauth package uses this ambiguity to help you encode names from many contexts. In this manual we refer to three classes of names, shown below. Section 2.2 shows how these classes are implemented.

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 implements that principle as a central part of its design. In each class there is a required name and optional name elements.[2] Other naming systems can be adapted to these general categories, *e.g.*, Icelandic, Hungarian, etc.

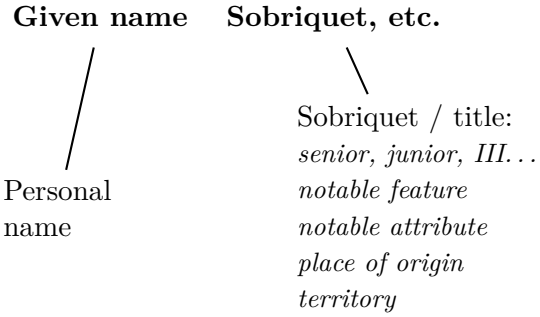
1. Western name:



2. Eastern name:



3. Ancient name:



[2] 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.

Choose Features

Choose Formatting

`mainmatter`	**Start with "main-matter names" and formatting hooks (page 7).**
`frontmatter`	Start with "front-matter names" and hooks.
`alwaysformat`	Use only respective "first use" formatting hooks.
`formatAKA`	Format names declared by `\AKA` like other main- and front-matter names.

The default `mainmatter` option and the `frontmatter` option enable two different systems of name use and formatting. They are mutually exclusive. `\NamesActive` starts the main matter system when `frontmatter` is used. See Section 2.5.9.

The `alwaysformat` option forces "first use" hooks globally. The `formatAKA` option affects "first-use" formatting and `\AKA` (Section 2.8.1). Both `alwaysformat` and `formatAKA` can be used with either `mainmatter` or `frontmatter`.

Enable/Disable Indexing

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

These apply only to the `nameauth` package macros. The default `index` option enables name indexing right away. The `noindex` option disables the indexing of names until `\IndexActive` enables it. **Both `noindex` and `\IndexInactive` suppress index tags.** See Section 2.9.1.

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 with NFSS or `makeindex` and its analogues. Seldom would one change this option unless a document is designed to work with both `makeindex` and `xindy`. See Section 2.9.4.

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*, 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.**
2. Section 2.4.1 shows how one can use `\ShowComma` with the `nocomma` option and `\NoComma` with the `comma` option to get per-name results.

Capitalize Entire Surnames

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

This only capitalizes names printed in the body text. English standards usually do not propagate typographic changes into the index.

Still, you can use this package with non-English conventions. You can add, *e.g.*, uppercase or small caps in surnames, formatting them also in the index. See also Sections 2.5.6 and 2.10.8 The simplified interface aids the embedding of control sequences in names. 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 all name forms in "smart" reverse order.
`allrevcomma`	Print all names in "Surname, Forenames" order, meant for Western names.

These options, along with th macros in Sections 2.4.2 and 2.5.3, assume that you are using long name references. Usually, the best approach is to use `\RevName` and `\RevComma` with `\Name*` and equivalents on a per-name basis.

See Section 2.4.2 for related macros to control name reversing by section or by name. Reversing can have unwanted results with short name references or ancient/medieval names.

So-called "last-comma-first" lists of names via `allrevcomma` and the macros `\ReverseCommaActive` and `\RevComma` (Section 2.5.3) are *not* the same as the `comma` option. They are designed for Western names.

Name Post-Processing

Sections 2.5.6, 2.5.10, and 2.10.6ff. explain this topic in greater detail, but are more advanced regarding technical details. Post-processing does not affect the index and comes after "syntactic formatting" (control sequences always present in the nameauth macro arguments).[3]

As of version 2.4, what was "typographic formatting" has become a generalized concept of "name post-processing" via these hook macros:

- `\NamesFormat` formats first uses of main-matter names.
- `\MainNameHook` formats subsequent uses of main-matter names.
- `\FrontNamesFormat` formats first uses of front-matter names.
- `\FrontNameHook` formats subsequent uses of front-matter names.

Sections 2.5.10 and 2.10.6ff. offer substantially more complex possibilities for such hooks.[4] By default, they do nothing.

English typography has been the default design choice for this package. Still, one can use German, French, and similar standards, which I group as "Continental." Sections 2.5.6 and 2.10.8 have more on the topic, as well as the use of sort tags in Section 2.9.4. Continental standards format surnames only, both in the text and in the index. This conflicts with some deliberately ambiguous name forms in nameauth.[5] To get formatting in the index one must add it in the macro arguments. The simplified interface aids this process. Continental users may have to implement their own capitalization features; see Sections 2.5.4 and 2.10.8.

Formatting Attributes

`smallcaps`	Set the first use of an entire name in small caps.[6]
`italic`	Set the first use of an entire name in italic.
`boldface`	Set the first use of an entire name in boldface.
`noformat`	**Do not define a default format.**

The options that assign a font change are intended for "quick" solutions based on English typography. They change only `\NamesFormat`, the macro that formats first instances of names in the main matter.

Versions 2.5 and onward assign no default formatting to names.

---

[3] This package was designed with type hierarchies in mind, although it has become more flexible. See Robert Bringhurst, *The Elements of Typographic Style*, version 3.2 (Point Roberts, Washington: Hartley & Marks, 2008), 53–60.

[4] I drew some inspiration from the typography in 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.

[5] Ancient, Eastern, and suffixed name forms have the same pattern.

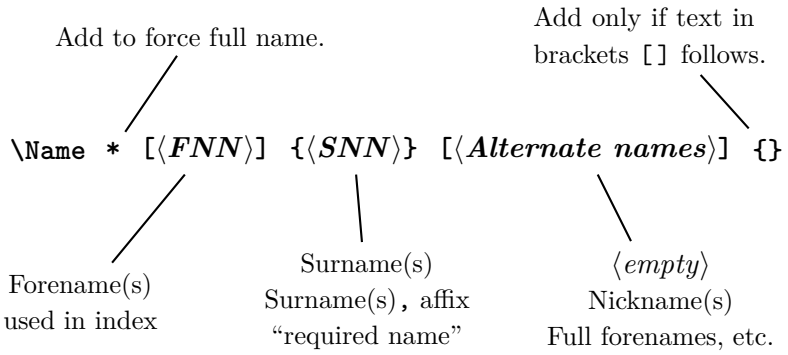
[6] Many users will not be affected by these changes. Many prefer the `noformat` option. If you did use the default option in the past, you can recover that behavior with the `smallcaps` option.

2.2 Quick Start Guide

2.2.1 Main Interface

Here we see how nameauth implements the classes of names in Section 1.2, using \Name as an example (see Section 2.3). The required name for each class is the ⟨*Surname(s)*⟩ argument (⟨*SNN*⟩). Optional name arguments include ⟨*Forename(s)*⟩ (⟨*FNN*⟩), ⟨*Alternate names*⟩ and ⟨*affix*⟩.

Western Names



Usual forms:

Forenames and affixes are in optional arguments and comma-delimited suffixes so they can drop in the text. Yet they always are included for consistent index entries.

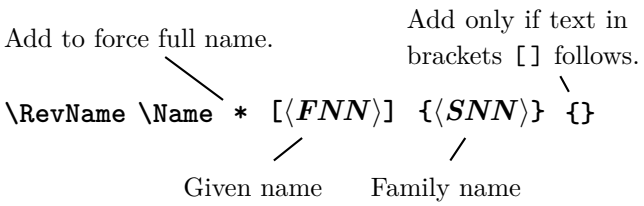
\Name[⟨*FNN*⟩]{⟨*SNN*⟩} . . . . . \Name[George]{Washington}
\Name[⟨*FNN*⟩]{⟨*SNN, affix*⟩} . . . . . \Name[John David]{Rockefeller, II}

When using ⟨*Alternate names*⟩, the ⟨*FNN*⟩ argument must have a name in it. 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]

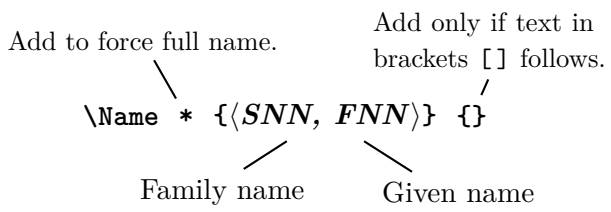
Western names cannot use the older syntax for affixes; see Sections 2.2.3 and 2.4.1.

Eastern Names in the Text, Western Index Entry



Technically, these are Western name forms without affixes. The reversing macros, such as \RevName (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 conflicts with both comma-delimited forms and the old syntax.

Eastern Names in the Text, Eastern Index Entry



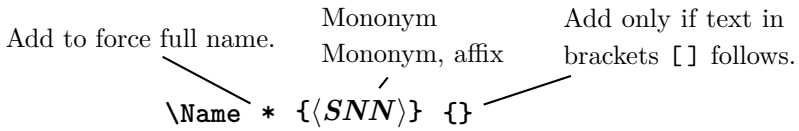
Usual form:

The striking feature of Eastern, ancient, and medieval name forms is the use of the comma-delimited suffix for the personal name. These forms have the same syntax, but different meanings. For example, you would use `\RevName` only with the Eastern 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 syntax for Eastern names is `\Name{`⟨*SNN*⟩`}[`⟨*FNN*⟩`]`, retained for backward compatibility (Section 2.2.3).

Ancient Names



Usual form:

These forms are used for royalty and ancient figures. They have one or more personal names that may or may not have suffixes.

`\Name{`⟨*SNN*⟩`}` . . . . . `\Name{Aristotle}`
`\Name{`⟨*SNN, affix*⟩`}` . . . . . `\Name{Elizabeth, I}`
. . . . . `\Name{Attila, the Hun}`

The older syntax takes the form `\Name{`⟨*Mononym*⟩`}[`⟨*affix*⟩`]`. Cf. Section 2.2.3.

Many macros in the main interface form a pattern, *e.g.*:

⟨*prefix macros*⟩[7]	`\Name`	⟨*arguments*⟩
⟨*prefix macros*⟩	`\Name*`	⟨*arguments*⟩
⟨*prefix macros*⟩	`\FName`	⟨*arguments*⟩
	`\IndexName`	⟨*arguments*⟩
	`\ForgetName`	⟨*arguments*⟩
	`\SubvertName`	⟨*arguments*⟩
	`\PretagName`	⟨*arguments*⟩ ⟨*sort key*⟩
	`\TagName`	⟨*arguments*⟩ ⟨*tag*⟩
	`\UntagName`	⟨*arguments*⟩
	`\ExcludeName`	⟨*arguments*⟩

[7] The ⟨*prefix macros*⟩ include one or more of `\CapThis`, `\AccentCapThis`, `\CapName`, `\RevName`, `\RevComma`, `\ShowComma`, `\NoComma`, and `\KeepAffix`.

2.2.2 Simplefied Interface

nameauth

The `nameauth` environment can replace `\Name, \Name*,` and `\FName` with shorthands. This aids consistency and brevity. Still, one must use all the other macros of the main interface. Both interfaces interoperate with each other.

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.[8] 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[9]
  \< ⟨cseq2⟩ & ⟨FNN⟩ & ⟨SNN, affix⟩   & >                    Western
  \< ⟨cseq3⟩ & ⟨FNN⟩ & ⟨SNN⟩          & ⟨Alt. names⟩ >      W. nickname[10]
  \< ⟨cseq4⟩ & ⟨FNN⟩ & ⟨SNN, affix⟩   & ⟨Alt. names⟩ >      W. nickname
  \< ⟨cseq5⟩ &       & ⟨SNN⟩          & >                    ancient/mono
  \< ⟨cseq6⟩ &       & ⟨SNN, affix⟩   & >                    royal/ancient
  \< ⟨cseq7⟩ &       & ⟨SNN, FNN⟩     & >                    Eastern[11]
  \< ⟨cseq8⟩ &       & ⟨SNN⟩          & ⟨FNN/affix⟩ >        old syntax[12]
\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*.
- Extra spaces in each `&`-delimited field are stripped, as is also the case in the main interface (Section 2.5.7).
- 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*⟩.

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

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

[10] 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 2.8.1.

[11] "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.

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

This example of the `nameauth` environment uses a fairly complete set of names, apart from some special cases. See how one can choose to line up the ampersands with extra spaces or not. Following that are examples of shorthand use in the body text. One may compare the forms in the text with the index.

```
\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}
```

Below, "non-native" Eastern name forms are shown with a dagger (†). Please see Section 2.4.2 to avoid pitfalls with the reversing macros. We show first uses of names in italic type.

Western:
`\Wash` . . . . . *George Washington*
`\LWash` . . . . . George Washington
`\Wash` . . . . . Washington
`\SWash` . . . . . George
`\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
`\SJRII` . . . . . John David

Nicknames: (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 / Mononym
`\Aris` . . . . . *Aristotle*
`\Aris` . . . . . Aristotle

Medieval/Royal:
`\Eliz` . . . . . *Elizabeth I*
`\Eliz` . . . . . Elizabeth
`\Attil` . . . . . *Attila the Hun*
`\Attil` . . . . . 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 2.5.4, 2.5.5, and 2.9.4 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 index tagging or pre-tagging names, the ⟨*Alternate names*⟩ field has no effect on index tags because it appears only in the text. `\JRIV` and `\JayR` need only one tag, as do `\Lewis` and `\Jack`:

```
\TagName[John David]{Rockefeller, IV}{⟨something⟩}
\TagName[Clive Staples]{Lewis}{⟨something⟩}
```

2.2.3 Older Syntax

An older syntax remains for backward compatibility with early versions of nameauth. The old syntax limits the use of `\AKA` and its derivatives. Section 2.12 and other places offer cautions about the old syntax. 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. Because that is the case, the final field changes its meaning from ⟨*Alternate names*⟩ to ⟨*affix*⟩ or ⟨*FNN*⟩. That means the final field will appear in the index. With this example, we have:

`\Name{Henry}[VIII]`	Henry VIII
`\Name{Henry}[VIII]`	Henry
`\Name{Chiang}[Kai-shek]`	Chiang Kai-shek
`\Name{Chiang}[Kai-shek]`	Chiang
`\Dagb`	Dagobert I
`\Dagb`	Dagobert
`\CapName\Yosh`	YOSHIDA Shigeru
`\CapName\RevName\LYosh`	Shigeru YOSHIDA

Additional Notes

`\Name{Henry}[VIII]` (old syntax) will share name occurrences, tags, and index entries with `\Name{Henry, VIII}` (new syntax), as we see below. Yet we do not recommend the random admixture of these forms.

```
\NameAddInfo{Henry}[VIII]{ (\emph{Defensor Fidei})}
...
\Name*{Henry, VIII}\NameQueryInfo{Henry, VIII}
```
Henry VIII (*Defensor Fidei*)

Avoid blending old and new syntax forms together in one name reference. `\Name{Henry, VIII}[Tudor]` prints "Henry VIII Tudor" and "Henry" in the body text and "Henry VIII Tudor" in the index. Sections 2.7 and 2.9.5 offer more elegant solutions that add "Tudor" either as a text tag or an index tag.

2.3 Naming Macros

2.3.1 `\Name` and `\Name*`

`\Name` `\Name*` This macro generates an index entry, a name in the body text, and another index entry for that name in case it spans a page break. `\FName` (next page) handles index entries in the same manner. 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 2.11. 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. `\Cicero` prints M.T. Cicero and Cicero. The preferred way to get alternate names is `\LCicero[Marcus Tullius]`: Marcus Tullius Cicero.

2.3.2 Forenames: \FName

\FName
\FName*

\FName and its synonym \FName* print just forenames, but only in subsequent name uses.[13] They are intended for Western-style names. The syntax is:

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

This macro prints a full name, not a short name, when a name is first used. That prevents a first-name reference before a person has been introduced. Section 2.6.2 has more on this. By design, \FName *never* prints Eastern personal names, so that ancient names also work (cf. Section 2.11). 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.

[13] 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. Other name types include royal, medieval, and Eastern names:

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

One cannot use the old syntax with the Hammerstein example. If you tried to use `\Name[Oskar]{Hammerstein}[II]` you would get "II Hammerstein." Western names with suffixes must use the new, comma-delimited syntax. Also, one must use comma-delimited suffixes to cross-reference all name forms with `\AKA`. For a full description see Section 2.8.1.

`\KeepAffix` Put `\KeepAffix` before `\Name` or `\AKA` if a ⟨*SNN, affix*⟩ pair is split: "Louis XIV" by a line break or page break. `\KeepAffix\Name*{Louis, XIV}` prevents that break by inserting a non-breaking space between ⟨*SNN*⟩ and ⟨*affix*⟩ (or ⟨*SNN*⟩ and ⟨*Eastern FNN*⟩) in the body text, but not in the index. Spaces between multiple names in each name field are not affected. This works with all name types handled by `nameauth`, including the old syntax.

`\ShowComma` If you do not want to use the `comma` option, `\ShowComma` gets the same results on a per-name basis for Western names while using the default `nocomma`. There is no affect on index entries. For example:

`\Name*[Louis]{Gossett, Jr.}`	Louis Gossett Jr.
`\ShowComma\Name*[Louis]{Gossett, Jr.}`	Louis Gossett, Jr.

`\NoComma` If you do wish to use the `comma` option, `\NoComma` can suppress a comma between the surname and affix on a per-name basis. For example, we simulate the `comma` option for the two tables below:

`\Name*[Louis]{Gossett, Jr.}`	Louis Gossett, Jr.
`\NoComma\Name*[Louis]{Gossett, Jr.}`	Louis Gossett Jr.

This might be useful when making a comma-reversed form (Section 2.5.3) under the `comma` option, where you do not want the form ⟨*SNN*⟩`,` ⟨*affix*⟩`,` ⟨*FNN*⟩:

`\RevComma\Name*[Louis]{Gossett, Jr.}`	Gossett, Jr., Louis
`\RevComma\NoComma\Name*[Louis]{Gossett, Jr.}`	Gossett Jr., Louis

One can switch freely among name forms, but remember that consistent use of `\ShowComma` and `\NoComma` is important when implementing name forms. The simplified interface can help with that.

2.4.2 Eastern Names

The reversing macros assume that you will be using long-name references, especially with "native" Eastern names. Please bear this in mind.

non-native

The nameauth package offers "non-native" and "native" ways to handle romanized Eastern names. The "non-native" form really is a Western name (and it is indexed as such) that is made to look Eastern in the body text:

`\RevName\Name*[`⟨*Eastern FNN*⟩`]{`⟨*Eastern SNN*⟩`}`

The index entry of this name form looks like ⟨*SNN*⟩, ⟨*FNN*⟩ (including the comma). This type of entry is a Western form. Pick non-native Eastern names when you need to have Western name forms in the index entries.

non-native

In contrast, there are two ways to make "native" Eastern name forms, which are indexed as such and appear in Eastern name order in the body text:

`\Name{`⟨*Eastern SNN, Eastern FNN*⟩`}`
`\Name{`⟨*Eastern SNN*⟩`}[`⟨*Eastern FNN*⟩`]` (old syntax)

The index entry of this name form looks like ⟨*SNN*⟩ ⟨*FNN*⟩ (no comma). This type of entry bears similarity with ancient and medieval forms. Pick native Eastern names when you want to use Eastern forms in the index.

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

In addition to the class options (Section 2.1), the macros `\ReverseActive` and `\ReverseInactive` toggle reversing on a larger scale, while `\RevName` is used once per name. The reverse output mechanism is designed to reverse full names only. Nevertheless, it does not force full names. Results vary, based on the type of Eastern name forms being used. Non-native forms are shown by a dagger (†):

	unchanged	`\RevName`
`\LKonoe`†	Fumimaro Konoe†	Konoe Fumimaro†
`\Konoe`†	Konoe†	Konoe†
`\LYamt`	Yamamoto Isoroku	Isoroku Yamamoto
`\Yamt`	Yamamoto	Isoroku

A striking feature above is that reversing a short native Eastern form will produce a personal name. This seems bad, especially when `\SYamt` *does not* produce a first name "Yamamoto," while `\RevName\Attil` does. Yet `\RevName\Attil` can be used to get only the affix in Attila "the Hun" instead of using the preferred method of text and index tags (Sections 2.7 and 2.9.5).[14]

Creating "last-comma-first" listings by surname (Section 2.5.3) only makes sense with Western names and maybe non-native Eastern names, but not with native Eastern names or ancient forms.

`\global`

Please note that `\ReverseActive` and `\ReverseInactive` can be used explicitly as a pair. They also can be used singly within an explicit scope, where the effects cease after leaving that scope. Use `\global` to force a global effect.

[14] Name reversing is a design point that really needs review.

Consider this list of Japanese music artists. All the name references are long names (`\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. Old-syntax forms have a double dagger(‡):

	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`

The nameauth package allows one to capitalize entire family names in the body text while keeping them in standard English form in the index. This capitalization is designed to work with Eastern name forms. 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 prefix macros, *e.g.*:

`\CapName\RevName\Name*{`⟨*SNN, Affix*⟩`}`.

`\global`

Both `\AllCapsActive` and `\AllCapsInactive` have the same local restrictions as the other state-changing macros. Use `\global` to force a global effect.

In the example below, names in Western order, then non-native Eastern order are marked with a dagger (†). All other names are in native Eastern, then Western order. Old-syntax forms have a double dagger(‡):

	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

Capitalization and reversing precede post-processing. 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 Sections 2.5.6 and 2.10.8.

2.5 Other Naming Topics

Language-Related Issues

2.5.1 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. Use `\PretagName` 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
Normal text:	E. B. White

2.5.2 Hyphenation

In English, some names come from other cultures. These names can break badly with standard hyphenation. The simplified interface trivializes the insertion of optional hyphens in names, but such hyphens must be used consistently:

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

...\IndexName[Johann]{Bier\-mann}
```

One also can fix bad breaks with the babel or polyglossia packages. This moves the solution from "quick and dirty" to elegant. We force a bad break: John Strietelmeier. We can stop that using babel:

```
\newcommand\de[1]{\foreignlanguage{ngerman}{#1}}
\de{\Name*[John]{Strietelmeier}}
```

2.5.3 Listing by Surname

`\ReverseCommaActive` `\ReverseCommaInactive` `\RevComma` `\global`

The reversing macros `\ReverseCommaActive`, `\ReverseCommaInactive`, and `\RevComma` let us reorder long Western names (via `\Name*` and the like). The first two are broad toggles, while the third works on a per-name basis.

Both `\ReverseCommaActive` and `\ReverseCommaInactive` have the same local restrictions as the other state-changing macros unless you use `\global.` Eastern, medieval, and royal names do not work with these macros.[15]

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

[15] Name reversing with commas is a design point that really needs review.

2.5.4 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*. This subsection deals mainly with accented names in English contexts.

`\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.

It is a good idea to put `~` or `\nobreakspace` between particles and surnames to avoid bad breaks. This also prevents `\CapThis` from eating the space between a one-character particle and the surname (Section 2.10.4).

The Continental style for surnames (Sections 2.5.6 and 2.10.8) does not work with the capitalization macros. For names like Catherine DE' MEDICI use `\Name[Catherine]{\textsc{de'~Medici}}` and, instead of the capping macros, `\textsc{De'~Medici}\IndexName[Catherine]{\textsc{de'~Medici}}`. Otherwise you will have to modify the examples from Section 2.10.8. Those examples can help you implement an alternate method of capitalization.

`\AccentCapThis`

With `pdflatex` and `inputenc`, use `\CapThis` when the first character of the particle is `A-z` (basic Latin). Use `\AccentCapThis` when the first character is extended Latin or other Unicode (see Section 2.10.4). Otherwise, with `pdflatex` `\CapThis` will fail if an extended Unicode character is the first letter of a particle.

For example, *L'* and *d'* are two separate glyphs each, while *Ľ* and *ď* are one Unicode glyph each. Even with `xelatex` and `lualatex`, you would put a non-breaking space between the particle and the name because the particle is only one character. With `pdflatex` you also must use `\AccentCapThis`.

Medieval names present some interesting difficulties, often based on the expected standards of the context in which they are used:

```
\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}
```

The medieval forms Thomas à Kempis and Thomas use the particle as the first part of an affix. Please do not confuse the medieval forms with the Western forms. Otherwise you will get similar names with different index entries.[16] Many people still use medieval affixes as Western surnames: "À Kempis."[17]

[16] 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.

[17] Ethnocentric treatment of names still occurs in academic literature. This is due to simplicity in work flow or conformity to name authorities. The nameauth package will accommodate those approaches, even if I personally disagree with them.

The Western form is `\AccentCapThis\KempW`. One might run into problems unless one considers the following caveats, as Section 2.10.4 explains:

- `\CapThis\KempW` halts execution with `Argument of \UTFviii@two@ octets has an extra \` when using `latex` or `pdflatex` and NFSS.
- `\AccentCapThis\Name[Thomas]{à Kempis}` produces "Thomas ÀKempis" (space removed). Instead, use `\AccentCapThis\Name[Thomas]{à~Kempis}`.
- With `latex` or `pdflatex` `\AccentCapThis` fails for particles like `lé` — use `\CapThis` in that case to avoid breaking the *second* character.
- `\AccentCapThis\Soto` gives DESoto, but only with `latex` or `pdflatex`. Only use it with accented first letters.

All this being said, the publisher's way of handling names may differ from the standard way. This package allows for such variations. Yet even some publishers fall afoul of confusion regarding name forms.[18] Developing a good rapport with the publisher will help you apply this package to the company's style.

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`. See Section 2.10.4 for more details.

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 2.11.2.

2.5.5 Accented Names

For names that contain accented characters, using `xelatex` or `lualatex` with `xindy` (`texindy`) is recommended. Section 2.10.5 shows how you can work with multiple engines using the same document.

If the leading character of ⟨*SNN*⟩ is accented and lowercase (usually only in a particle), then you must use `\AccentCapThis` if you are using NFSS. Sections 2.5.4 and 2.10.4 give more details about `\CapThis` and `\AccentCapThis`.

Accented characters act like control sequences. In NFSS use `\PretagName` for all names with extended Unicode characters (Sections 2.9.4 and 2.10.4).[19]

Unicode characters and "regular" control sequences are not interchangeable. The example below shows that the names are all long (thus, different). Were the control sequences identical, the names would be long, then short:

`\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

See Section 2.10.4 on how to add additional Unicode glyphs to the default set under NFSS, inputenc, and fontenc. One may use expandable control sequences in names (thanks Robert Schlicht). Also, you can use `\edef` and `\noexpand` in names (Section 2.10.8, thanks Patrick Cousot).

[18] An example of a true error is the index entry "Yat-sen, Sun" (as if Sun were a forename) in Immanuel Geiss, *Personen: Die biographische Dimension der Weltgeschichte*, Geschichte Griffbereit vol. 2 (Munich: Wissen Media Verlag, 2002), 720. Still, the six-volume set is good.

[19] This is true especially in NFSS while using `makeindex`. With `xindy` one can make custom sorting alphabets that are more powerful than `\PretagName`.

2.5.6 Non-English Format

See Sections 2.5.9, 2.5.10, and 2.10.8 in addition to this section. We place this section here because it has a language-based element.

Continental small caps

By default, name post-processing only affects the text. Also by default, you cannot post-process Continental formatting because it occurs in both the text and the index via control sequences in the `nameauth` macro arguments. That usually requires `\PretagName` for index sorting (Section 2.9.4). Section 2.10.8 explains how to implement post-processing via `\noexpand` and other steps.

Here we have a very basic attempt at Continental formatting of the surname in both the body text and the index. One cannot alter this formatting:

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

You get Greta Garbo, then Garbo — even in the front matter. In order to get an unformatted reference to the name "Garbo," one must type something like ``Garbo''\IndexName[Greta]{\textsc{Garbo}}.

As with accented names, different control sequences produce different names. `\Name[\normalfont{Greta}]{\textsc{Garbo}}` looks exactly like name above, but it will create a different index entry and have different first / subsequent uses.

A comma delimiter will split the macro argument into a root and an affix. This could halt typesetting and generate an error regarding unbalanced braces. Avoid this by formatting the name and suffix separately. Here we manually format Continental and Eastern names that contain root / affix pairs:

```
\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.

The old syntax cannot be used with Thurston Howell III. For FUKUYAMA Takeshi, the old syntax would be:

```
\PretagName{\uppercase{Fukuyama}}[Takeshi]{Fukuyama Takeshi}
\begin{nameauth}
  \< OFukuyama & & \uppercase{Fukuyama} & Takeshi >
\end{nameauth}
```

`\LOFukuyama` produces FUKUYAMA Takeshi. `\OFukuyama` yields FUKUYAMA. This form works with the new syntax form and is not distinct from it. It is the same as `\Name{\uppercase{FUKUYAMA}}[Takeshi]`. The caps macros will not work on it, apart from the examples in Section 2.10.8.

2.5.7 Fault Tolerance

The nameauth package protects against some typing errors. Adding extra spaces *normally* does not produce unique names. The simplified interface also is fault-tolerant with spaces. The variations below illustrate this point:

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.

2.5.8 Detecting Punctuation

In Western names, affixes like "Jr." (junior), "Sr." (senior), "d. J." (*der Jüngere*), and "d. Ä." (*der Ältere*) can collide with the full stop in a sentence. `\Name`, `\FName`, and `\AKA` detect this in a name and gobble the subsequent full stop as needed:

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.

Grouping tokens can frustrate this: `{\Name*[Martin Luther]{King, Jr.}}.` produces "Martin Luther King Jr.." (two periods). Enclosing `{Jr.}` within braces or a macro will do the same. If you must format the affix, leave the period outside: `\Name[Martin Luther]{\textsc{King}, \textsc{Jr}.}` Cf. Section 2.5.6.

2.5.9 Format: Systems

`\NamesActive` `\NamesInactive`

Using the `frontmatter` option or `\NamesInactive` causes the naming macros to use the front matter formatting hook until `\NamesActive` switches the macros to the independent main matter formatting hooks. Additionally, two independent systems of names are created: front-matter names and main-matter names.

`\global`

Please note that these two macros can be used explicitly as a pair. They also can be used singly within an explicit scope, where the effects cease after leaving that scope. Use `\global` to force a global effect.

By default, these two systems of names differ only between first and subsequent uses. Here we change the first-use formatting hooks (Section 2.5.10) to show a greater distinction:

```
\let\OldFormat\NamesFormat
\let\OldFrontFormat\FrontNamesFormat
\renewcommand*\NamesFormat{\scshape}
\renewcommand*\FrontNamesFormat{\bfseries}
```

Here we switch to the "front matter" mode:

```
\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:

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

This illustrates the independent systems or "species" of names. This is most useful when you want to format names one way in the regular body text but another way somewhere else. In footnotes, for example, we could redefine `\NamesFormat` to create custom formatting:

```
\makeatletter
\let\@oldfntext\@makefntext%                save original macro
\long\def\@makefntext#1{%
  \renewcommand*\NamesFormat{\scshape}\@oldfntext{#1}}
\let\@makefntext\@oldfntext% just in case
\makeatother
```

The problem is that John Maynard Keynes in the text could affect formatting in the footnotes.[20] Instead of editing footnotes as needed by inserting `\ForgetName,` there is a different solution that uses the front-matter system:

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

Your footnotes do not affect the main body text now.[21] We change footnotes back to normal with the following:

```
\makeatletter%
\let\@makefntext\@oldfntext%
\makeatother
```

At this point we could restore the formatting hooks, but we will do that in the next section. We would do that by scoping or with:

```
\let\NamesFormat\OldFormat
\let\FrontNamesFormat\OldFrontFormat
```

[20] You get Keynes instead of John Maynard Keynes.

[21] We have **John Maynard Keynes**, then Keynes.

2.5.10 Format: Hooks

There are two kinds of formatting at work:

1. **Syntactic Formatting:** For the English default, this includes what the reversing and capitalizing macros do. For the alternatives, it includes any control sequences embedded in the macro arguments.
2. **Name Post-Processing:** This happens in the hook macros after a name has been parsed into the final form it will take in the text.

`\NamesFormat`
`\FrontNamesFormat`
`\MainNameHook`
`\FrontNameHook`

Starting with version 2.5, the "main-matter" and "front-matter" systems are fully qualified and independent systems of formatting and first/subsequent name use. The main-matter system uses `\NamesFormat` to post-process first occurrences of names and `\MainNameHook` to post-process subsequent uses. The front-matter system uses `\FrontNamesFormat` to post-process first occurrences of names and `\FrontNameHook` to post-process subsequent uses. The `alwaysformat` option causes every name to be typeset as a "first-use."

Front Matter

`\Name[Albert]{Einstein}`	**Albert Einstein**
`\Name[Albert]{Einstein}`	Einstein
`\Name{Confucius}`	**Confucius**
`\Name{Confucius}`	Confucius
`\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

Main Matter

`\Name[Albert]{Einstein}`	Albert Einstein
`\Name[Albert]{Einstein}`	Einstein
`\Name{Confucius}`	Confucius
`\Name{Confucius}`	Confucius
`\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

Below we simulate the `alwaysformat` option by manipulating the package internals. We see that this option does not force long name forms. After the examples, we reset the formatting hooks.

- Using `alwaysformat` in the front matter will produce: **Albert Einstein**, then **Einstein**; **Confucius**, then **Confucius**.
- Using `alwaysformat` in the main matter will produce: Marcus Tullius Cicero, then Cicero; Charles the Bald, then Charles.

2.6 Name Decisions

2.6.1 Testing Decisions

The macros in this section permit conditional text that depends on the presence or absence of a name. These macros use `\If...` because they differ from regular `\if` expressions. The following macros affect conditional branching: `\Name`, `\Name*`, `\FName`, `\PName`, `\AKA`, `\AKA*`, `\ForgetName`, `\SubvertName`, and `\ExcludeName`. One might consider some uses:

- a book where one includes a margin paragraph, mini-bio, footnote, or other information when a name is first introduced
- a game book where you have to pick a game path
- a presentation that can change if certain names are present
- the "text tag" features in Section 2.7
- conditional comments using the comment, pdfcomment, and similar packages

If one uses these macros inside other macros or passes control sequences to them, the expansion of control sequences can create false results (see *The TEXbook*, 212–15). To get around those problems, consider using the following:

- Use token registers to retrieve the arguments.
- Regulate expansion with `\expandafter`, `\noexpand`, etc.
- That affects accented characters in `pdflatex`/NFSS.

See Sections 2.10.7 and 2.10.8 for related ideas about tokens and expansion. Using `\tracingmacros`, `\show`, or `\meaning` can help you.

`\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 get "I have not met Bob" from the following example because we have yet to invoke `\Name[Bob]{Hope}`:

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

Please note that this test is not affected by the use of `\IndexName`. Since we have encountered Johann Andreä, we get "I met Johann" with a similar example:

```
\IfMainName[Johann]{Andreä}{I met Johann}%
  {I have not met Johann}
```

`\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, based on Section 2.5.9, we see that "Carnap is both" a formatted and unformatted name with the following test:

```
\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}}
```

Please refer to Sections 2.6.2 and 2.10.2 to understand the scope and operation of main- and front-matter names.

`\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 additional ⟨*excluded*⟩ branch in order to detect those names excluded from indexing by `\ExcludeName` (Section 2.9.7).

A "*see*-reference" name is printed in the body text but only exists as a cross-reference created by `\AKA` and `\AKA*`. First, in the text we see "Jay Rockefeller," `\AKA[John David]{Rockefeller, IV}[Jay]{Rockefeller}`. Next, we have the following example:

```
\IfAKA[Jay]{Rockefeller}%
  {\LJRIV\ has an alias}%
  {\LJRIV\ has no alias}%
  {\LJRIV\ is excluded}
```

This gives us "John David Rockefeller IV has an alias." 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 we should test that completely:

```
\IfAKA[⟨FNN⟩]{⟨SNN⟩}[⟨alt. names⟩]%
{⟨true; it is a pseudonym⟩}%
{%
  \IfFrontName[⟨FNN⟩]{⟨SNN⟩}[⟨alt. names⟩]%
    {\IfMainName[⟨FNN⟩]{⟨SNN⟩}[⟨alt. names⟩]%
      {⟨both⟩}%
      {⟨front⟩}%
    }%
    {\IfMainName[⟨FNN⟩]{⟨SNN⟩}[⟨alt. names⟩]%
      {⟨main⟩}%
      {⟨does not exist⟩}%
    }%
}%
{⟨excluded⟩}
```

Here we test for a name used with `\ExcludeName` (Section 2.9.7) 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.

One use for this macro is to get around the safeguards of `\FName`. To ensure formatting consistency:

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

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 on page 26 above that gave us the answer, "Carnap is both?" 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.[22]

`\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 work globally.

After `\LocalNames,` if you are in a "front matter" section (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. Section 2.10.2 offers a long example.

2.7 "Text Tags"

Sections 2.9.5 and 2.9.6 deal with similar tagging features in the index. "Text tags" differ from index tags because they are not printed automatically with every name managed by `nameauth`. Section 2.10.7 offers additional solutions that use the macros in this section.

Instead of "text tags," perhaps one should think about "name information database entries." The macros in this section are named accordingly. We retain the "text tag" language for simplicity.

`\NameAddInfo`

Text tags are independent of any other name conditionals, similar to index tags. This `\long` macro's syntax is:

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

For example, `\NameAddInfo[George]{Washington}{ (1732--99)}` will associate the text `␣(1732-99)` with the name `\LWash` George Washington. Note, however, that the tag did not print automatically with the name.

[22] 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.

\NameQueryInfo To retrieve the information in a text tag, one uses the name as a key to the corresponding information:

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

Thus, ‘‘\NameQueryInfo[George]{Washington}.’’ expands to “ (1732–99).” Notice the space at the beginning of the tag. This is intentional, as with index tags. Sections 2.9.5, 2.9.6, and 2.10.6ff. illustrate how this can permit tags like asterisks, daggers, and footnotes, such as one for Schuyler Colfax.[23] The source for that example looks like:

```
\NameAddInfo[Ulysses S.]{Grant}{ (president 1869--77)}%
\NameAddInfo[Schuyler]{Colfax}%
{\footnote{Seventeenth vice-president of the US during%
the first term (1869--73) of \Name[Ulysses S.]{Grant}%
\NameQueryInfo[Ulysses S.]{Grant}.}}
...
\Name[Schuyler]{Colfax}.\NameQueryInfo[Schuyler]{Colfax}
```

By using these text tag macros with the conditional macros, one can display information associated with a name based on whether or the name has occurred. As of version 2.4, this can be done either outside of \NamesFormat and the other general hooks or inside those macros.

\NameClearInfo \NameAddInfo will replace one text tag with another text tag, but it does not delete a text tag. That is the role of \NameClearInfo. The syntax is:

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

For example, \NameClearInfo[George]{Washington} will cause the macro ‘‘\NameQueryInfo[George]{Washington}’’ to produce nothing.

2.8 Name Variant Macros

2.8.1 \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*⟩]

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 exists.

\AKA prints the ⟨*Alt. FNN*⟩ and ⟨*Alt. SNN*⟩ fields in the body text. If the ⟨*Alt. names*⟩ field is present, \AKA prints the ⟨*Alt. names*⟩ and ⟨*Alt. SNN*⟩ arguments in the body text. The caps and reversing macros work with \AKA.

For Western alternate names, in the body text, \AKA* only prints either ⟨*Alt. FNN*⟩ or ⟨*Alt. names*⟩, if present. For non-Western alternate names, \AKA* prints either ⟨*Alt. SNN*⟩ or ⟨*Alt. names*⟩ if present.

[23] Seventeenth vice-president of the US during the first term (1869–73) of Ulysses S. Grant (president 1869–77).

Basic Operation

These are the possible main-name forms before the alternate name:

```
\AKA  [⟨FNN⟩]{⟨SNN⟩}          ...   Western
\AKA  [⟨FNN⟩]{⟨SNN, Affix⟩}   ...
\AKA          {⟨SNN⟩}         ...   Ancient
\AKA          {⟨SNN, Affix⟩}  ...
\AKA          {⟨SNN, FNN⟩}    ...   Eastern
```

These are the "better" alternate-name forms after the main name:

```
\AKA  ...  [⟨Alt. FNN⟩]{⟨Alt. SNN⟩}                          Western
\AKA  ...  [⟨Alt. FNN⟩]{⟨Alt. SNN⟩}[⟨Alt. names⟩]
\AKA  ...  [⟨Alt. FNN⟩]{⟨Alt. SNN, Affix⟩}
\AKA  ...  [⟨Alt. FNN⟩]{⟨Alt. SNN, Affix⟩}[⟨Alt. names⟩]
\AKA  ...             {⟨Alt. SNN⟩}                          Ancient
\AKA  ...             {⟨Alt. SNN, Affix⟩}
\AKA  ...             {⟨Alt. SNN, Alt. FNN⟩}                Eastern
```

These alternate-name forms use the old syntax:

```
\AKA  ...             {⟨Alt. SNN⟩}[⟨Alt. names⟩]            Eastern
\AKA  ...             {⟨Alt. SNN⟩}[⟨Alt. names⟩]            Ancient
```

See also Sections 2.9.5 and 2.10.3. The next example makes "alternate name" cross-references to Bob Hope, illustrating Western names:

`\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.

As with nicknames and `\Name`, the alternate form "Lester T. Hope" does not appear in the index, but only in the body text. The next example makes "alternate name" cross-references to Louis XIV, Lao-tzu, and Gregory I:

`\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

Formatting Alternate Names

In this simple example we redefine the the default system of formatting to illustrate what can happen under default formatting conditions:

```
\renewcommand*\NamesFormat{\scshape}
\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.

`formatAKA` "Jean the Fearless" usually receives no formatting because it is post-processed by `\MainNamesHook` in the main matter text and `\FrontNamesHook` in the front matter. The `formatAKA` option causes `\AKA` to use `\NamesFormat` and `\FrontNamesFormat`, but only for the first use in the whole document. This allows name exclusion to work and avoids possible errors with index cross-references. Using the `alwaysformat` option formats all names as first uses:

Illustrating `\AKA{Elizabeth, I}[Good Queen]{Bess}`
and `\AKA{Elizabeth, I}{Virgin Queen}`:

`formatAKA`

Front matter: *Elizabeth I* was known as "*Good Queen Bess.*" Queen Elizabeth also was known as the "*Virgin Queen.*"
Main matter: Elizabeth I was known as "Good Queen Bess." Queen Elizabeth also was known as the "Virgin Queen."

`alwaysformat`

Front matter: *Elizabeth I* was known as "*Good Queen Bess.*" Queen *Elizabeth* also was known as the "*Virgin Queen.*"
Main matter: Elizabeth I was known as "Good Queen Bess." Queen Elizabeth also was known as the "Virgin Queen."

The following annotated example shows a simple Continental style where the surname is always in small caps, both in the text and in the index:

1. Tag the names for proper sorting.

   ```
   \PretagName[Heinz]{\textsc{Rühmann}}{Ruehmann, Heinz}%
   \PretagName[Heinrich Wilhelm]{\textsc{Rühmann}}%
     {Ruehmann, Heinrich Wilhelm}%
   ```

2. "Heinz Rühmann" is the main name. `\AKA*` uses "Rühmann, Heinrich Wilhelm" as the index cross-reference and prints only "Heinrich Wilhelm" in the body text.

   ```
   \AKA*[Heinz]{\textsc{Rühmann}}%
     [Heinrich Wilhelm]{\textsc{Rühmann}} %
   ```

3. `\SubvertName` causes `\FName` to print the short version via the "subsequent-use" macro `\MainNameHook`.

   ```
   \SubvertName[Heinz]{\textsc{Rühmann}} %
   ```

4. `\FName` prints "Heinz."

   ```
   ``\FName[Heinz]{\textsc{Rühmann}}'' %
   ```

5. `\Name` prints "Rühmann." The small caps are syntactic, not typographic, because they are part of the argument to `\Name` itself.

   ```
   \Name[Heinz]{\textsc{Rühmann}} (7 March 1902--3 October %
   1994) was a German actor in over 100 films.
   ```

The resulting text is:

Heinrich Wilhelm "Heinz" Rühmann (7 March 1902–3 October 1994) was a German actor in over 100 films.

`\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}}
```

Keep in mind, however, that formatting manual index entries can present some caveats; see Section 2.9.4.

Cross-references generated by `\AKA` and `\AKA*` are meant only to be *see* references, never page entries. See also Section 2.12. 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.[24]

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, even though many people, including scholars, falsely use it as such. Section 2.5.4 talks about those unfortunate situations where one must use the Western form `\Name[Thomas]{Aquinas}`.

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

2.8.2 \PName

\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. Below we see the forms that \Pname can handle:

```
\PName[Mark]{Twain}[Samuel L.]{Clemens}
```
. . . . . Mark Twain (Samuel L. Clemens)
. . . . . Twain (Samuel L. Clemens)

```
\PName*[Mark]{Twain}[Samuel L.]{Clemens}[Sam]
```
. . . . . Mark Twain (Sam Clemens)

```
\PName{Voltaire}[François-Marie]{Arouet}
```
. . . . . Voltaire (François-Marie Arouet)
. . . . . Voltaire (François-Marie Arouet)

```
\PretagName{\textit{Doctor mellifluus}}{Doctor mellifluus}
\PName{Bernard, of Clairvaux}{\textit{Doctor mellifluus}}
```
. . . . . Bernard of Clairvaux (*Doctor mellifluus*)
. . . . . Bernard (*Doctor mellifluus*)

Like \AKA, \PName cannot use the old syntax {⟨*SNN*⟩}[⟨*FNN*⟩] for the main name, but it can do so for the alternate name.

\PName{William, I}{William, the Conqueror} gives William I (William the Conqueror). To limit possible confusion, avoid the old syntax in the alternate name: \PName{William, I}{William}[the Conqueror]. Nevertheless, that *does* work and will produce William (William the Conqueror). If \PName* to get William I (William the Conqueror).

Also choose forms like \PName{Lao-tzu}{Li, Er} "Lao-tzu (Li Er)" instead of \PName{Lao-tzu}{Li}[Er] "Lao-tzu (Li Er)." Both forms will work, but look real confusing when interchanged.

The form \PName{William, I}[William]{the Conqueror} will produce "William (William the Conqueror)" in the body text, but its index entry will be "the Conqueror, William *see* William I." This is a result of mixing medieval and Western forms.

2.9 Indexing Macros

2.9.1 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.

\global

Please note that these two macros can be used explicitly as a pair. They also can be used singly within an explicit scope, where the effects cease after leaving that scope. Use `\global` to force a global effect.

Index tags only work when indexing is active.

2.9.2 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 fairly effective workaround for `texindy` redefines `\textlatin` to produce the page number itself within a certain scope like:

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

2.9.3 `\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.

2.9.4 Index Sorting

The general practice for sorting with `makeindex -s` involves creating your own `.ist` file (pages 659–65 in *The Latex Companion*). Otherwise the following form works with both `makeindex` and `texindy`: `\index{`⟨*sortkey*⟩`@`⟨*actual*⟩`}`

Basic Sorting (Makeindex)

`\PretagName` Since version 2.0, nameauth integrates this sort of index sorting automatically by using a "pretag." The syntax is:

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

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

```
\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 Jan 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 2.8.1.

Although the `\PretagName` macro might look similar to the the other tagging macros, its use and scope is quite a bit different:

- 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."

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

Extra Spaces and Sorting

Under NFSS, extended Unicode characters expand to add one or two spaces after control sequences. See `\indexentry` and `\item` entries in your `idx` and `ind` files. For example, `ä` becomes `\IeC␣{\"a}` (one space added) and `Æ` becomes `\IeC␣{\AE␣}` (two spaces added). Both `xelatex` and `lualatex` (using fontspec) avoid these issues by handling the characters natively.

NFSS: `\index{Fußball}` → `\indexentry{Fu\IeC␣{\ss␣}ball}{`⟨*page*⟩`}`

fontspec: `\index{Fußball}` → `\indexentry{Fußball}{`⟨*page*⟩`}`

cseq: `\index{Fu\ss ball}` → `\indexentry{Fu\ss␣ball}{`⟨*page*⟩`}`

A macro with the general form below, similar to `\IndexName`, will add two spaces after *some* control sequences. Those spaces only affect index sorting, not appearance. Remember this when using manual index entries with nameauth:

```
\newcommand\IndexExample[1]{%
    \protected@edef\argument{#1}\index{\argument}}%
```

`\IndexExample{\textsc{football}}` →
`\indexentry{\textsc␣␣{football}}{`⟨*page*⟩`}`

`\index{\textsc{football}}` →
`\indexentry{\textsc{football}}{`⟨*page*⟩`}`

2.9.5 \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, as can other package features. Here \TagName causes the nameauth indexing macros to append "␣pope" to the index entries for Gregory I and Leo I:

```
\TagName{Leo, I}{, pope}                  (in the preamble)
\TagName{Gregory, I}{, pope}
...
\Name*{Leo, I} was known as               Leo I was known as
\AKA{Leo, I}{Leo}[the Great].             Leo the Great.
...
\Name{Gregory, I} ``\AKA*{Gregory, I}%    Gregory "the Great,"
{Gregory}[the Great],'' also was          also was a major pope.
a major pope.
```

Tags are literal text that can be daggers, asterisks, and even specials. For example, all fictional names in the index of this manual are tagged with an asterisk. One must add any desired spacing to 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.

2.9.6 \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.[25]

2.9.7 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*⟩`]`

Consider the following example, where 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 2.9.1).

[25] 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.

2.10 Longer Examples

2.10.1 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.

2.10.2 \LocalNames

As mentioned previously in Section 2.6.2, both \ForgetName and \SubvertName usually affect both main-matter and front-matter names. This default behavior can be quite helpful. Nevertheless, there are cases where it is undesirable. This section shows \Localnames and \Globalnames in action, limiting the behavior of the "tweaking macros" to either the main or front matter.

We begin by defining a macro that will report to us whether a name exists in the main matter, front matter, both, or none:

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

Next we create a formatted name in the "main matter":

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

Now we switch to "front-matter" text and create a name. We use \global with \NamesInactive in order to ignore any local scoping environments:

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

We now have two names. They look and behave the same, but are two different "species" with independent first and subsequent uses. We use \Localnames to make \ForgetName and \SubvertName work with only the front-matter species. Then we "forget" the front-matter name:

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

Next we "subvert" the front-matter name to "remember" it again and switch to the main section, again using \global to ignore scoping. Now \ForgetName and \SubvertName are working with the main-matter species.

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

We forget the main-matter name and additionally reset the default behavior so that \ForgetName and \SubvertName work with both species:

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

Finally, we forget everything. Even though we are in a main-matter section, the front-matter control sequence goes away:

```
\ForgetName[Charlie]{Chaplin}
\CheckChuck                                            none
```

2.10.3 Tips for \AKA

- [⟨*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 2.11.2.
- Reversing works, *e.g.*,
  \RevComma\AKA...: Hope, Leslie Townes
  \RevName\AKA... : 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}.The results are 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.

  – Follow ``Iron Mike'' with \IndexName[Mike]{Tyson} and do whatever you want in the text. This may be the easiest solution.
  – 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 2.8.1. This is the best solution in terms of how nameauth is designed.
  – 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."[26] 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.

[26] 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.

2.10.4 Unicode and NFSS

The following subset of extended Latin Unicode characters are available "out of the box" using NFSS, inputenc, and fontenc:

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

Some of these characters expand differently, which can affect index sorting. For example, ä becomes `\IeC␣{\"a}` and Æ becomes `\IeC␣{\AE␣}`. 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}
```

Although `\newunicodechar{ā}{\=a}` allows `\Name{Ghazāli}` to generate Ghazāli, one must be careful with control sequences like `\=a` fail when using `makeindex` and `gind.ist`. For example, the ltxdoc class, with `gind.ist`, turns the default "actual" character `@` into `=`. 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.

Such issues with `gind.ist` are not the only concerns one must have about NFSS, inputenc, and fontenc when using Unicode. Although the manner in which glyphs are handled is quite powerful, it also is fragile. Any TEX macro that partitions its argument without using delimiters can break Unicode under NFSS. Consider the following examples with `\def\foo#1#2#3\relax{<#1#2><#3>}`:

Argument	Macro	Result
`abc`	`\foo abc\relax`	<ab><c>
`{æ}bc`	`\foo {æ}bc\relax`	<æb><c>
`\aebc`	`\foo \ae bc\relax`	<æb><c>

The arguments in the last example always put `c` in `#3`, with the first two glyphs in `#1#2`. Now here is where things get tricky:

Argument	Macro	Engine	Result
æbc	\foo æbc\relax	xelatex	<æb><c>
æbc	\foo æbc\relax	lualatex	<æb><c>
æbc	\foo æbc\relax	pdflatex	<æ><bc>

In both `xelatex` and `lualatex` you get the same results as the previous table, where `c` is in `#3` and the first two glyphs are in `#1#2`. However, using `latex` or `pdflatex` with `inputenc` and `fontenc` causes `æ` by itself to use `#1#2`.

Without digging into the details of font encoding and NFSS, we can say in simple terms that `æ` 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 }`. This is not just specific to `nameauth`.

Using `\CapThis` can trigger this kind of error when the *first* character of the ⟨*SNN*⟩ field is an extended-Latin or similarly accented or extended Unicode character. Using `\AccentCapThis` can trigger this kind of error when the *second* character of the ⟨*SNN*⟩ field is a similarly accented or extended character.

LaTeX also removes spaces between macro arguments in a manner that one should remember:

Argument	Macro	Result
a b c	\foo a b c\relax	<ab>< c>
ab c	\foo ab c\relax	<ab>< c>
a bc	\foo a bc\relax	<ab><c>
abc	\foo abc\relax	<ab><c>

Notice that if a space exists between the first two arguments,the space gets gobbled between the first two arguments, but retained in the third. This pertains to the way that LaTeX allows for spaces after control sequences and tries to fetch the undelimited `#1#2`. Since `#3` terminates the argument list, it gets "everything else." Nor would using `\obeyspaces` and `\ignorespaces` always get the desired result without a certain degree of complexity.

Here is why using explicit spacing macros with one-character particles when using `\CapThis` and `\AccentCapThis` helps fix the issue of gobbled spaces, and why non-breaking spaces are preferred:[27]

Argument	Macro	Result
a~bc	\foo a~bc\relax	<a ><bc>
a\nobreakspace bc	\foo a\nobreakspace bc\relax	<a ><bc>
a\space bc	\foo a\space bc\relax	<a ><bc>

See also Sections 2.5.4 and 2.5.5.

[27] Given that you would not want a bad break between a particle and a name.

2.10.5 LaTeX Engines

The nameauth package tries to work with multiple languages and typesetting engines. The following preamble snippet from this manual illustrates how that can be done:[28]

```
\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
```

This arrangement worked best for this manual, which is tested with all of the LaTeX engines above. This example is not meant to be the only possible way to check which engine you are using and how to set things up.

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
```

[28] A similar version of this example is in `examples.tex`, collocated with this manual.

2.10.6 Hooks: Intro

Margin Paragraphs

Before we get to the use of text tags and name conditionals in name formatting, we begin with an intermediate example to illustrate that something more complex can occur in `\NamesFormat`. Here we put the first mention of a name in boldface, along with a marginal notation if possible:[29]

```
\let\OldFormat\NamesFormat%
\renewcommand*\NamesFormat[1]%
  {\textbf{#1}\ifinner\else
  \marginpar{\raggedleft\scriptsize #1}\fi}
...
\let\NamesFormat\OldFormat%
```

Changes to `\NamesFormat` should not rely merely on scoping rules to keep them "local" but should be changed and reset explicitly, or else odd side effects can result, especially with more exotic changes to `\NamesFormat`. We now use the example above in a sample text:

```
\PretagName{Vlad, Ţepeş}{Vlad Tepes}% for accented names

\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.

Now again we have reverted to the original form of `\NamesFormat` and we get 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 and avoid the old syntax, lest errors bite! The simplified interface greatly helps one to avoid this.

You cannot re-enter `\Name` or `\AKA` by calling them within `\Namesformat`, `\FrontNameHook`, or `\MainNameHook`, as the next example shows:

```
\renewcommand*\MainNameHook[1]%
{%
  {#1}%
  \IndexInactive%
  \Name{foo}\AKA{bar}{baz}%
  \IndexActive%
}
```

Calling, *e.g.*, `\Wash` produces Washington, without foo, bar, or baz. `\Name` and `\AKA` expand to nothing. Version 2.4 of `nameauth` prevents stack-overflows both in this case and if you called the naming macros as their own arguments. `\Name{foo\Name{bar}}` would produce "FOO" in the text and "fooBAR" in the index. As you see, these cases are to be avoided.

[29] A similar version of this example is in `examples.tex`, collocated with this manual.

2.10.7 Hooks: Life Dates

We can use name conditionals (Section 2.6.1) and text tags (Section 2.7) to add life information to names when desired.

\if@nameauth@InName
\if@nameauth@InAKA

The example \NamesFormat below adds a text tag to the first occurrences of main-matter names. It uses internal macros of \@nameauth@Name. To prevent errors, the Boolean values \@nameauth@InName and \@nameauth@InAKA are true only within the scope of \@nameauth@Name and \AKA respectively.

\@nameauth@toksa
\@nameauth@toksb
\@nameauth@toksc

This package makes three token registers available to facilitate using the name conditional macros as we do below. Using these registers allows accented names to be recognized properly. In \AKA the token registers are copies of the *last* three arguments, corresponding to the pseudonym. Nevertheless, they have the same names as the registers in \@nameauth@Name because they work the same way and may be easier to use this way.

We assume that we will not be using the alwaysformat option, meaning that we only call this hook once for a first use:[30]

```
\newif\ifNoTextTag%       allows us to work around \ForgetName
\let\OldFormat\NamesFormat%                    save the format
\makeatletter%                                access internals
\renewcommand*\NamesFormat[1]%
{%
  \let\ex\expandafter%                           reduce typing
  \textbf{#1}%
  \if@nameauth@InName%              do only in \@nameauth@Name
    \ifNoTextTag%                    true branch disables tags
    \else%                                    take false branch
      \ex\ex\ex\ex\ex\ex\ex\NameQueryInfo%
      \ex\ex\ex\ex\ex\ex\ex[%
      \ex\ex\ex\the\ex\ex\ex\@nameauth@toksa\ex\ex\ex]%
      \ex\ex\ex{\ex\the\ex\@nameauth@toksb\ex}%
      \ex[\the\@nameauth@toksc]%
    \fi
  \fi
  \if@nameauth@InAKA%                             do only in \AKA
    \ifNoTextTag\else
      \ex\ex\ex\ex\ex\ex\ex\NameQueryInfo%
      \ex\ex\ex\ex\ex\ex\ex[%
      \ex\ex\ex\the\ex\ex\ex\@nameauth@toksa\ex\ex\ex]%
      \ex\ex\ex{\ex\the\ex\@nameauth@toksb\ex}%
      \ex[\the\@nameauth@toksc]%
    \fi
  \fi
  \global\NoTextTagfalse%                 reset tag suppression
}
\makeatother%
```

The example above prints tags by default in the false path of \NoTextTag, while suppressing them in the true path.

[30] A similar version of this example is in examples.tex, collocated with this manual.

Before we can refer to any text tags, we must create them. For teaching purposes I will "lie" (sorry) about the tag used for "Atatürk" until later.

```
\NameAddInfo[George]{Washington}{ (1732--99)}%
\NameAddInfo[Mustafa]{Kemal}{ (1881--1938)}%
\NameAddInfo{Atatürk}{ (in 1934, a special surname)}%
```

We begin using the modified `\NamesFormat` under normal conditions:

```
\Wash held office 1789--97. No tags appear with \Wash. %
First use, dates suppressed:%
\NoTextTagtrue\ForgetName[George]{Washington}\Wash.

\Name[Mustafa]{Kemal} was granted the name%
\AKA[Mustafa]{Kemal}{Atatürk}. We mention%
\AKA[Mustafa]{Kemal}{Atatürk} again.
```

George Washington (1732–99) held office 1789–97. No tags appear with Washington. First use, dates suppressed: **George Washington**.

Mustafa Kemal (1881–1938) was granted the name Atatürk. We mention Atatürk again.

Notice that the text tag for Atatürk did not print. That is because `\AKA` usually only calls the "subsequent use" hooks. Therefore we simulate the `formatAKA` option and `\ForgetName` Washington and Kemal:

George Washington (1732–99) held office 1789–97. No tags appear with Washington. First use, dates suppressed: **George Washington**.

Mustafa Kemal (1881–1938) was granted the name **Atatürk** (in 1934, a special surname). We mention Atatürk again.

Here we see that the tag is printed because `formatAKA` allows `\NamesFormat` to be called for the first use of Atatürk.

Now we `\let` the first-use macro in the front matter be the same as that in the main matter and see what we get in the front matter via `\NamesInactive.` Again we simulate the `formatAKA` option and `\ForgetName` Washington and Kemal:

```
\NamesInactive
\let\OldFrontFormat\FrontNamesFormat
\let\FrontNamesFormat\NamesFormat
```

George Washington (1732–99) held office 1789–97. No tags appear with Washington. First use, dates suppressed: **George Washington**.

Mustafa Kemal (1881–1938) was granted the name **Atatürk** (in 1934, a special surname). We mention Atatürk again.

I have not been quite honest above. Since I wanted to simulate multiple first uses of `\AKA,` which can print a tag only once unless you use the `alwaysformat` option, I inserted kerns into the instances of `\AKA[Mustafa]{Kemal}{Atatürk}` in the example paragraphs above:

```
\NameAddInfo{\kern0pt Atatürk}{ (a special surname granted 1934)}
\NameAddInfo{Atatürk\kern0pt}{ (a special surname granted 1934)}
\NameAddInfo{Atatürk}{ (a special surname granted 1934)}
```

Because the names were different in each paragraph even though they looked the same, I prevented the names with kerns from appearing in the index via `\IndexInactive` and `\IndexActive.`

Please remember to reset the formatting, if needed:

```
\let\NamesFormat\OldFormat
\let\FrontNamesFormat\OldFrontFormat
```

Here is a summary of the general behavior within the naming and hook macros and the kinds of decisions you might consider:

1. In `\@nameauth@name` and `\AKA`:
   (a) Parse name arguments. Save an unexpanded copy of each relevant name argument in a token register.
   (b) Check for a control sequence based on them.
   (c) Enter a decision route based on the result. Yes means the name exists. No means it does not. The decision route engages the Boolean values governing formatting.
   (d) Generate the index and print forms of the name. Create the index entry from the former and pass the latter onward to the format switching macro. Based on the Boolean values governing formatting, that macro calls either the first-use or subsequent-use hooks respectively in the main matter or the front matter.
2. In the post-processing hooks:
   (a) Normally you do nothing and exit, or make a local font change and exit. You could do more complex tasks like discarding the text output of the naming macros, then parsing and displaying the name parameters differently, based on the token registers.
   (b) You also can make more than one independent check for a control sequence based on the name arguments saved in the token registers. This permits some fairly complex actions based on both the Boolean values and the control sequences themselves.
   (c) Thus your decision route could turn into a tree or a set of relationships among a number of names.
   (d) Print the form of the name as it was passed, or possibly do something else altogether.
   (e) **If you invoke `\@nameauth@name` and `\AKA` from within the hooks, they will do nothing.**
3. In `\@nameauth@name` and `\AKA`:
   (a) Generate the control sequence that says the name exists.
   (b) Make a second index entry in case of page breaks, clean up and reset any Boolean flags, and exit.

2.10.8 Hooks: Continental and Caps

Continental Format

For implementing Continental systems of name formatting, in addition to the basic methods already discussed, here we see how one can have the small caps consistently in the surnames for the first uses and the index, yet have a normal font for subsequent references in the body text.

The big difference between this example and the optional caps is that here the font change is optional, while with the caps example, the font change is always present and the caps are optional. And yes, it is possible to do both an optional font change and optional caps.

The literal text of this example goes in the document preamble. We add a Boolean value that turns small caps on or off, depending on the context. Please do not change this directly.

```
\newif\ifSC
\SCtrue%                          We want small caps in the index
```

`\DoFormat` is the key to this approach. We want a control sequence that will expand differently, depending on the state of the Boolean value above.

```
\def\DoFormat#1{%
  \ifSC \textsc{#1}%                  Format small caps if true
  \else #1%                                 Do nothing if false
  \fi}
```

`\noexpand` is another vital piece of the solution. If one does not use this, all sorts of errors will arise. See how the formatting always is in the ⟨*SNN*⟩ field.

```
\begin{nameauth}
  \< JQA & John Quincy & \noexpand\DoFormat{Adams} & >
  \< Aeths & & \noexpand\DoFormat{Æþelstan} & >
  \< Chas & & \noexpand\DoFormat{Charles}, I & >
  \< Cao & & \noexpand\DoFormat{Cao}, Cao & >
  \< JRIII & John David & \noexpand\DoFormat{Rockefeller}, III & >
  \< SDJR & Sammy & \noexpand\DoFormat{Davis},\noexpand\DoFormat{Jr}. & >
\end{nameauth}
```

Now we must ensure that these names are sorted properly in the index. See again how the formatting must be present:

```
\PretagName[John Quincy]%
  {\noexpand\DoFormat{Adams}}{Adams, John Quincy}
\PretagName{\noexpand\DoFormat{Æþelstan}}{Aethelstan}
\PretagName{\noexpand\DoFormat{Charles}, I}{Charles 1}
\PretagName{\noexpand\DoFormat{Cao}, Cao}{Cao Cao}
\PretagName[John David]{\noexpand\DoFormat{Rockefeller}, III}%
  {Rockefeller, John David, III}
\PretagName[Sammy]%
  {\noexpand\DoFormat{Davis}, \noexpand\DoFormat{Jr}.}%
  {Davis, Sammy, Jr.}
```

We save the hook macros if we want to recall them.

```
\let\OldNamesFormat\NamesFormat
\let\OldFrontNamesFormat\FrontNamesFormat
\let\OldFrontHook\FrontNameHook
\let\OldMainHook\MainNameHook
```

We do not need to redefine either `\NamesFormat` or `\FrontNamesFormat` because we already set the default to be small caps. We redefine `\MainNameHook` and `\FrontNameHook` in order to *suppress* formatting in subsequent uses of names.

The new implementation of `\MainNameHook` follows. We incorporate those parts of `\AKA` and `\@nameauth@Name` that print name arguments in the text.[31] Please bear in mind that this example is quite long.

```
\makeatletter%
\renewcommand*\MainNameHook[1]%
{%
  \let\ex\expandafter%
  \SCfalse%
```

Above we set the small caps Boolean to false. Now we have to "redo" name parsing in order to get the different format. We expand the naming macros from the token registers set by the parent naming macro.

```
\protected@edef\arga{\ex\trim@spaces\ex{\the\@nameauth@toksa}}%
\protected@edef\argb{\ex\trim@spaces\ex{\the\@nameauth@toksb}}%
\protected@edef\testb{\ex\@nameauth@Root\ex{\the\@nameauth@toksb}}%
\protected@edef\argc{\ex\trim@spaces\ex{\the\@nameauth@toksc}}%
```

`\Space` (below) is defined by the parent naming macro. Before we print an appropriate version of the pseudonym or name in the text, we determine the removable suffix, if any, and the reversed forms.

```
\ifx\argb\testb
  \protected@edef\Suff{\@empty}%
  \let\Reversed\argb%
  \let\SNN\argb%
  \let\Short\argb%
\else
  \protected@edef\Suff{\ex\@nameauth@Suffix\ex{%
    \the\@nameauth@toksb}}%
  \protected@edef\Reversed{\Suff\Space\testb}%
  \protected@edef\SNN{\testb\Space\Suff}%
   \if@nameauth@RevThis
     \let\Short\Suff%
   \else
     \let\Short\testb%
   \fi
\fi
```

`\AllCapsActive`, `\CapName`, and the `allcaps` package option are ignored here because they are incompatible with the Continental approach. Only the reversing macros will be usable.

[31] A similar version of this example is in `examples.tex`, collocated with this manual.

If the parent naming macro is \AKA and its variants, print an appropriate version of the pseudonym in the text.

```
\if@nameauth@InAKA
  \ifx\arga\@empty
    \ifx\argc\@empty
      \if@nameauth@RevThis \Reversed \else \SNN \fi
    \else
      \if@nameauth@AltAKA \argc%
      \else
        \if@nameauth@RevThis \ex\argc\ex\space\SNN%
        \else \ex\SNN\ex\space\argc%
        \fi \fi \fi
  \else
    \ifx\argc\@empty \let\FNN\arga%
    \else \let\FNN\argc%
    \fi
    \if@nameauth@AltAKA \FNN%
    \else
      \if@nameauth@RevThis \ex\SNN\ex\Space\FNN%
      \else \ex\FNN\ex\space\SNN%
      \fi \fi \fi
\else
```

If the main naming engine is the parent macro, print an appropriate version of the name in the text.

```
  \ifx\arga\@empty
    \ifx\argc\@empty
      \if@nameauth@FullName
        \if@nameauth@RevThis \Reversed \else \SNN \fi
      \else \Short \fi
    \else
      \if@nameauth@FullName
        \if@nameauth@RevThis \ex\argc\ex\space\SNN%
        \else \ex\SNN\ex\space\argc%
        \fi
      \else
        \if@nameauth@RevThis \argc \else \Short \fi
      \fi \fi
  \else
    \ifx\argc\@empty \let\FNN\arga%
    \else \let\FNN\argc%
    \fi
    \let\Short\FNN%
    \if@nameauth@FullName
      \if@nameauth@RevThis \ex\SNN\ex\Space\FNN%
      \else \ex\FNN\ex\space\SNN%
      \fi
    \else
      \if@nameauth@FirstName \Short \else \testb \fi
    \fi \fi \fi
}
```

Now we let `\FrontNameHook` be the same as `\MainNameHook`, and we are ready to see the Continental approach in action.

```
\let\FrontNameHok\MainNameHook
\makeatother
```

Main Matter

First	Next	Long	Short
John Quincy Adams	Adams	John Quincy Adams	John Quincy
John David Rockefeller III	Rockefeller	John David Rockefeller III	John David
Æþelstan	Æþelstan	Æþelstan	Æþelstan
Charles I	Charles	Charles I	Charles
Cao Cao	Cao	Cao Cao	Cao

Front Matter

First	Next	Long	Short
John Quincy Adams	Adams	John Quincy Adams	John Quincy
John David Rockefeller III	Rockefeller	John David Rockefeller III	John David
Æþelstan	Æþelstan	Æþelstan	Æþelstan
Charles I	Charles	Charles I	Charles
Cao Cao	Cao	Cao Cao	Cao

- Punctuation detection still works: Sammy Davis Jr. Then we have Davis.
- `\RevComma\LJQA` yields Adams, John Quincy. All the reversing macros work.
- Using the main interface requires some extra typing. For example, `\ForgetName[John Quincy]{\noexpand\DoFormat{Adams}}\JQA` results in John Quincy Adams.

Normally `\AKA` will not format alternate names. However, if we use the `formatAKA` option we can refer to Cao Cao as Mengde, and again Mengde. We get that with:

```
\PretagName{\noexpand\DoFormat{Mengde}}{Mengde}
\AKA{\noexpand\DoFormat{Cao}, Cao}{\noexpand\DoFormat{Mengde}}
```

If you want to suppress formatting altogether in the front matter, make the following change: `\let\FrontNamesFormat\MainNameHook`

Front Matter (hooks changed; names forgotten)

First	Next	Long	Short
John Quincy Adams	Adams	John Quincy Adams	John Quincy
John David Rockefeller III	Rockefeller	John David Rockefeller III	John David
Æþelstan	Æþelstan	Æþelstan	Æþelstan
Charles I	Charles	Charles I	Charles
Cao Cao	Cao	Cao Cao	Cao

When needed, we can use `\let` to restore the hooks to their old values. Please note, however, that the index entries will contain small caps, regardless of how we change the hook macros for the document text.

Caps within Formatting

This method of redesigning the hooks must be used if you want to capitalize a name that is otherwise formatted in the index. All we need to do is modify the example above. We begin in similar fashion, but add a few extra bits:[32]

```
\newif\ifItal%                              Flag to trigger italics
\newif\ifFirstCap%                             Flag to trigger caps
\newif\ifInHook%          But only with a name in the body text
\Italtrue%                              We want italics in the index
\def\DoFormat#1{%
  \ifItal\textit{#1}\else#1\fi
}
\def\CP#1{%
  \ifInHook
    \ifFirstCap\uppercase{#1}\else#1\fi%
  \else
    #1%
  \fi
}
\newcommand*\CapMe{\FirstCaptrue}
```

Like the previous example on Continental formatting, you would put the formatting in the naming macro arguments. `\CP` can fail using NFSS and Unicode, so take care! We have changed the meaning of `\DoFormat` in this section. If we were to use any of the names from the Continental example above, they would now be formatted italic and produce different index entries that also would be italicized instead of small caps.

```
\PretagName[Pierre-Jean]{\noexpand\DoFormat{\noexpand\CP{d}e Smet}}%
  {de Smet, Pierre-Jean}
\begin{nameauth}
 \< deSmet & Pierre-Jean & \noexpand\DoFormat{\noexpand\CP{d}e Smet}} & >
\end{nameauth}
```

We take the code of `\MainNameHook` above and redefine it to be `\NamesFormat`. We change the example above to read:

```
\renewcommand*\MainNameHook[1]{%
  \let\ex\expandafter%
  \InHooktrue%                    cap in the text, not the index
```

Leave the rest of the example intact and change the last part of the macro:

```
  \InHookfalse\global\FirstCapfalse%
}\makeatother%
```

You need to have `\global` before `\FirstCapfalse` to ensure that the change persists beyond the scope. By design, the hooks occur within a group in the naming functions.

[32] A similar version of this example is in `examples.tex`, collocated with this manual.

If you want italics all the time, just `\let` all the hooks be `\NamesFormat`. If you want to follow the Continental style similar to the example above, you do:

```
\let\FrontNamesFormat\NamesFormat
\renewcommand*\MainNameHook[1]%
  {\Italfalse\NamesFormat{#1}\Italtrue}
\let\FrontNameHook\MainNameHook
```

Now we show how the formatting hooks work in the body text. One can check the index to see that it is formatted with italics and is consistent.

Main Matter

First	Next	Long	Short
`\deSmet` Pierre-Jean *de Smet*	`\deSmet` de Smet	`\LdeSmet` Pierre-Jean de Smet	`\SdeSmet` Pierre-Jean

Front Matter

First	Next	Long	Short
`\deSmet` Pierre-Jean *de Smet*	`\deSmet` de Smet	`\LdeSmet` Pierre-Jean de Smet	`\SdeSmet` Pierre-Jean

The capitalized version `\CapMe\deSmet` is De Smet. If we let `\MainNameHook` be the same as `\NamesFormat` we can get *De Smet*. The index entries will be consistent for all the variations in the text.

Of course, one could go a step further and integrate this capitalizing mechanism with the `\CapThis` and `\AccentCapThis` macros. That, however, goes beyond the scope of this example. You would have to insert the capitalization bits from `\@nameauth@name`. Or you could test for the latex engine and do everything in the hook, and redefine `\CapThis`.

Also, remember to restore the macro hooks if they should not persist for the entire document, or else you will get unwanted results:

```
\let\NamesFormat\OldNamesFormat%
\let\FrontNamesFormat\OldFrontNamesFormat%
\let\FrontNameHook\OldFrontHook%
\let\MainNameHook\OldMainHook%
```

2.10.9 Full Redesign

Assuming that redefining hooks and adding control sequences is insufficient to your task, you could modify the core naming macros and hook those modifications back into the nameauth package without needing to continuously track and patch the style file itself.

\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⟩}%
\newcommandx*\MyLName[3][1=\@empty, 3=\@empty]%
  {⟨Long name⟩\@nameauth@FullNamefalse}%
\newcommandx*\MyFName[3][1=\@empty, 3=\@empty]%
  {⟨Short name⟩\@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*⟩, ⟨*Long name*⟩, and ⟨*Short name*⟩.

\global

Like `\NamesFormat`, the other hook macros, and many of the state-changing and triggering macros in this package, these naming macros can be redefined or used locally within a scope without making global changes to the document unless you specifically use `\global`.

Here we show that the macros `\NameauthName, \NameauthLName,` and `\NameauthFName` have reverted back to their original forms. Now `\Silly` and `\Name[No Particular]{Name}` produce No Particular Name and Name.

2.11 Naming Pattern Reference

2.11.1 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

`\Name{Ptolemy}[I Soter]` keeps the number with the affix. To keep the number with the name, use `\Name{Ptolemy I}[Soter]`. 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

The old syntax `\Name{Antiochus}[IV Epi\-phanes]` is functionally equivalent to `\Name{Antiochus, IV Epi\-phanes}`. Consider also how you might handle affixes and sobriquets:

- `\Name{Antiochus, IV Epi\-phanes}` yields Antiochus IV Epiphanes and Antiochus in the body text; "Antiochus IV Epiphanes" in the index.
- `\Name{Antiochus~IV, Epi\-phanes}` yields Antiochus IV Epiphanes and Antiochus IV in the body text and "Antiochus IV Epiphanes" in the index.
- `\Name{Antiochus, IV}` yields Antiochus IV and Antiochus in the text. Use `\NameAddInfo`, `\NameQueryInfo`, and `\IndexTag` to handle "Epiphanes." We prefer, use, and recommend this way.

2.11.2 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.

2.12 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 2.9.4.
- 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.
- 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 2.8.1
- 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 an xref `[`⟨*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. `\ShowComma` toggles the second and `\NoComma` toggles the third. Each instance of `\Name` and `\AKA` reset `\@nameauth@ShowComma`.

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

Toggle Formatting

`\NamesActive` and `\NamesInactive` or the `mainmatter` and `frontmatter` options make `\@nameauth@MainFormat` either true or false, which switches between the main and front matter hooks. `\@nameauth@AKAFormat` permits `\AKA` to call the first-use hooks. Otherwise it will call only the subsequent-use hooks.

```
\newif\if@nameauth@MainFormat
\newif\if@nameauth@AKAFormat
```

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
```

Syntactic Formatting

`\@nameauth@FullName` is true in any case where a long name reference is desired. `\@nameauth@FirstName` disables full-name references and causes only Western forenames to be displayed. `\@nameauth@AltAKA` is toggled respectively by `\AKA` and `\AKA*` to print a longer or shorter name.

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

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
```

This Boolean value is triggered by `\CapThis` and reset by `\Name` and `\AKA`.

```
\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
```

`\KeepAffix` toggles the value below, which causes `\Name` and `\AKA` to use non-breaking spaces between a name and an affix, then reset the value.

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

This Boolean value is used for detection of double full stops at the end of a name.

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

Hook Triggers

`\@nameauth@FirstFormat` triggers the first-use hooks to be called; otherwise the second-use hooks are called. Additionally, `\@nameauth@AlwaysFormat` forces this true, except when `\@nameauth@AKAFormat` is false.

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

Who Called Me?

These values are true within `\Name` and `\AKA`, respectively. Otherwise they are false. They are used when one modifies the hook macros. See Sections 2.10.7ff.

```
\newif\if@nameauth@InAKA
\newif\if@nameauth@InName
```

As an aside, `\AKA` will invoke `\NamesFormat` / `\FrontNamesFormat` if the `alwaysformat` option is set. Otherwise it will invoke `\MainNameHook` / `\FrontNameHook`.

Stack Overflow Prevention

Here is the locking mechanism that prevents a stack overflow via recursive calls to `\Name` and `\AKA`. See Sections 2.10.7ff.

```
\newif\if@nameauth@Lock
```

3.2 Hooks

`\NamesFormat` Post-process "first" instance of final complete name form in text. See Sections 2.5.10 and 2.10.6ff. Called when both `\@nameauth@MainFormat` and `\@nameauth@FirstFormat` are true.

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

`\MainNameHook` Post-process subsequent instance of final complete name form in main-matter text. See Sections 2.5.10 and 2.10.6ff. Called when `\@nameauth@MainFormat` is true and `\@nameauth@FirstFormat` is false.

```
\newcommand*\MainNameHook{}
```

`\FrontNamesFormat` Post-process "first" instance of final complete name form in front-matter text. Called when `\@nameauth@MainFormat` is false and `\@nameauth@FirstFormat` is true.

```
\newcommand*\FrontNamesFormat{}
```

`\FrontNameHook` Post-process subsequent instance of final complete name form in front-matter text. Called when `\@nameauth@MainFormat` is false and `\@nameauth@FirstFormat` is false.

```
\newcommand*\FrontNameHook{}
```

`\NameauthName` Hook to create custom naming macros. Usually the three macros below have the same control sequence, but they need not do so if you want something different. See Section 2.10.9. Use at your own risk! Changing these macros basically rewrites this package.

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

`\NameauthLName` Customization hook called after `\@nameauth@FullName` is set true. See Section 2.10.9.

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

`\NameauthFName` Customization hook called after `\@nameauth@FirstName` is set true. See Section 2.10.9.

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

Name Argument Token Registers

These three token registers contain the current values of the name arguments passed to `\Name`, its variants, and the cross-reference fields of `\AKA`.

```
\newtoks\@nameauth@toksa%
\newtoks\@nameauth@toksb%
\newtoks\@nameauth@toksc%
```

These three token registers contain the current values of the name arguments in each line of the `nameauth` environment.

```
\newtoks\@nameauth@etoksb%
\newtoks\@nameauth@etoksc%
\newtoks\@nameauth@etoksd%
```

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@MainFormattrue}
\DeclareOption{frontmatter}{\@nameauth@MainFormatfalse}
\DeclareOption{formatAKA}{\@nameauth@AKAFormattrue}
\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,%
   noformat}
\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}
```

The etoolbox package is essential for bringing the modern functionality of ε-TEX in parsing and passing the name parameters, etc. Using xargs allows for the optional arguments to work in a fairly wide set of environments.[33] Using suffix facilitated macros like `\Name*`, although one might argue whether or not a "starred form" is the best approach, especially when suffix and xargs have some compatibility issues. Finally, trimspaces helps the fault tolerance of name arguments and ifluatex/ifxetex allow accented names to work on different LATEX engines.

3.4 Internal Macros

Name Control Sequence: Who Am I?

`\@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}
```

Core Name Parsing Operations

`\@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
}
```

[33] Early versions of this package used LATEX3 functionality that was powerful. Yet the naming macros broke in some cases, like in `\marginpar` and some other environments.

`\@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}}
```

Punctuation Detection

`\@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}
```

Name Hook Dispatcher

`\@nameauth@FmtName` The dispatcher invokes the appropriate formatting hooks, depending on the Boolean values for first use, subsequent use, and name type. The first set of tests enables or disables formatting within `\AKA`. The second set of tests handle all the other naming macros in the name and front matter. The hooks have a local scope.

```
\newcommand*\@nameauth@FmtName[1]%
{%
  \if@nameauth@InAKA
    \if@nameauth@AlwaysFormat
      \@nameauth@FirstFormattrue%
    \else
      \if@nameauth@AKAFormat\else\@nameauth@FirstFormatfalse\fi
    \fi
    \@nameauth@TestDot{#1}%
    \if@nameauth@MainFormat
      \if@nameauth@FirstFormat
        \bgroup\NamesFormat{#1}\egroup%
      \else
        \bgroup\MainNameHook{#1}\egroup%
      \fi
    \else
      \if@nameauth@FirstFormat
        \bgroup\FrontNamesFormat{#1}\egroup%
      \else
        \bgroup\FrontNameHook{#1}\egroup%
      \fi
    \fi
  \else
    \if@nameauth@AlwaysFormat\@nameauth@FirstFormattrue\fi
    \@nameauth@TestDot{#1}%
    \if@nameauth@MainFormat
      \if@nameauth@FirstFormat
        \bgroup\NamesFormat{#1}\egroup%
      \else
        \bgroup\MainNameHook{#1}\egroup%
      \fi
    \else
      \if@nameauth@FirstFormat
        \bgroup\FrontNamesFormat{#1}\egroup%
      \else
        \bgroup\FrontNameHook{#1}\egroup%
      \fi
    \fi
  \fi
}
```

Core Indexing Operations

\@nameauth@Actual This sets the "actual" character used by nameauth for index sorting.

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

\@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
}
```

Core Name Management Engine

\@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]%
{%
```

Prevent entering \@nameauth@Name via itself or \AKA. Both \@nameauth@Name and \AKA engage the lock. Calling these macros in their own parameters will create malformed output but should not halt program execution or overflow the stack. Calling these macros within the hook macros will simply cause them to exit.

```
  \if@nameauth@Lock\else
  \@nameauth@Locktrue%
  \@nameauth@InNametrue%
  \let\ex\expandafter%
```

Names occur in horizontal mode; we ensure that. Next we make copies of the arguments to test them and make parsing decisions. We also make token register copies of the current name args to be available for the hook macros.

```
\leavevmode\hbox{}%
\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}}%
\@nameauth@toksa\expandafter{#1}%
\@nameauth@toksb\expandafter{#2}%
\@nameauth@toksc\expandafter{#3}%
```

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 reversing is true, print a full name.

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

Create an index entry before we print the name.

```
\IndexName[#1]{#2}[#3]%
```

The code below handles non-breaking and regular spaces, as well as commas, in the text, which are inserted only for Western names.

```
\protected@edef\Space{\space}%
\if@nameauth@NBSP\protected@edef\Space{\nobreakspace}\fi
\ifx\arga\@empty\else
  \if@nameauth@AlwaysComma
    \protected@edef\Space{,\space}%
    \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
  \fi
  \if@nameauth@ShowComma
    \protected@edef\Space{,\space}%
    \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
  \fi
  \if@nameauth@NoComma
    \protected@edef\Space{\space}%
    \if@nameauth@NBSP\protected@edef\Space{\nobreakspace}\fi
  \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}}%
\ifx\testb\testbr
  \let\Suff\@empty%
  \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\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
```

Now we begin to format 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. If formatting is active, we handle the main matter names (code !MN). First we handle subsequent uses. We need `\expandafter` to enable the punctuation detection.

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

Here we handle first uses.

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

If formatting is inactive we process names in the front matter (code !NF for non-formatted). First handle subsequent uses.

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

Handle first uses of front-matter names.

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

This is the section that handles the old syntax for royal names and native Eastern names. 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@MainFormat
        \ifcsname\csbc!MN\endcsname
          \if@nameauth@FirstName
            \@nameauth@FullNamefalse%
          \fi
          \if@nameauth@FullName
            \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
          \@nameauth@FullNamefalse%
          \@nameauth@FirstNamefalse%
        \else
```

Handle first uses.

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

Handle first uses.

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

```
  \if@nameauth@RevThisComma
    \protected@edef\Space{,\space}%
    \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
  \fi
  \ifx\testc\@empty
    \let\FNN\arga%
  \else
    \let\FNN\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@MainFormat
  \ifcsname\csab!MN\endcsname
    \if@nameauth@FirstName
      \@nameauth@FullNamefalse%
      \let\PrintShort\FNN%
    \fi
    \if@nameauth@FullName
      \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
    \@nameauth@FullNamefalse%
    \@nameauth@FirstNamefalse%
  \else
```

Handle first uses.

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

Handle first uses.

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

Make another index entry.

```
  \IndexName[#1]{#2}[#3]%
```

Reset all the "per name" Boolean values.

```
  \@nameauth@Lockfalse%
  \@nameauth@InNamefalse%
  \@nameauth@FirstFormatfalse%
  \@nameauth@NBSPfalse%
  \@nameauth@DoCapsfalse%
  \@nameauth@Accentfalse%
  \@nameauth@AllThisfalse%
  \@nameauth@ShowCommafalse%
  \@nameauth@NoCommafalse%
  \@nameauth@RevThisfalse%
  \@nameauth@RevThisCommafalse%
```

Close the "locked" branch.

```
  \fi
```

Call the full stop detection.

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

3.5 User Interface Macros

Syntactic Formatting — Capitalization

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

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

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

```
\newcommand*\AccentCapThis%
  {\@nameauth@Accenttrue\@nameauth@DoCapstrue}
```

`\CapName` Capitalize entire name.

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

`\AllCapsInactive` Turn off global surname capitalization.

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

`\AllCapsActive` Turn on global surname capitalization.

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

Syntactic Formatting — Reversing

`\RevName` Reverse name order.

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

`\ReverseInactive` Turn off global name reversing.

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

`\ReverseActive` Turn on global name reversing.

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

Syntactic Formatting — Reversing with Commas

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

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

`\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}
```

Syntactic Formatting — Affixes

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

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

`\NoComma` Remove comma between name and suffix one time (with `comma` option)..

```
\newcommand*\NoComma{\@nameauth@NoCommatrue}
```

Typographic Formatting — Affixes

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

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

Typographic Formatting — Main Versus Front Matter

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

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

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

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

Name Occurrence Tweaks

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

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

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

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

Index Operations

\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}}
```

Main Naming Interface

\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}
```

Alternate Names

\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]%
{%
```

Prevent entering `\AKA` via itself or `\@nameauth@Name`.

```
  \if@nameauth@Lock\else
  \@nameauth@Locktrue%
  \@nameauth@InAKAtrue%
  \let\ex\expandafter%
```

Names occur in horizontal mode; we ensure that. Next we make copies of the arguments to test them and make parsing decisions. We also make token register copies of the current name args to be available for use within the hook macros.

```
  \leavevmode\hbox{}%
  \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\argd{\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}}%
  \@nameauth@toksa\expandafter{#3}%
  \@nameauth@toksb\expandafter{#4}%
  \@nameauth@toksc\expandafter{#5}%
```

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, which are inserted only for Western names.

```
\protected@edef\Space{\space}%
\if@nameauth@NBSP\protected@edef\Space{\nobreakspace}\fi
\ifx\argc\@empty\else
  \if@nameauth@AlwaysComma
    \protected@edef\Space{,\space}%
    \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
  \fi
  \if@nameauth@ShowComma
    \protected@edef\Space{,\space}%
    \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
  \fi
  \if@nameauth@NoComma
    \protected@edef\Space{\space}%
    \if@nameauth@NBSP\protected@edef\Space{\nobreakspace}\fi
  \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 `\@nameauth@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%
  \let\Suffb\@empty%
\else
  \protected@edef\Suffb{\@nameauth@Suffix{#2}}%
  \protected@edef\SNNb{\Shortb\space\Suffb}%
\fi
\ifx\testd\testdr
  \let\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\space\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 begin to index and format 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
          \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\Space{,\space}%
      \if@nameauth@NBSP
        \protected@edef\Space{,\nobreakspace}%
      \fi
    \fi
    \ifcsname\csd!PN\endcsname\else\@nameauth@FirstFormattrue\fi
    \if@nameauth@RevThis
      \ex\@nameauth@FmtName\ex{\Reversed}%
    \else
      \ex\@nameauth@FmtName\ex{\SNNd}%
    \fi
    \ifcsname\csd!PN\endcsname\else\csgdef{\csd!PN}{}\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
            \ifx\testa\@empty
              \@nameauth@Index{\csde}%
                {\ISNNd\space\arge|see{\SNNb}}%
            \else
              \ifx\Suffb\@empty
                \@nameauth@Index{\csde}%
                  {\ISNNd\space\arge|see{\SNNb,\space\arga}}%
              \else
                \@nameauth@Index{\csde}%
                {\ISNNd\space\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\Space{,\space}%
        \if@nameauth@NBSP
          \protected@edef\Space{,\nobreakspace}%
        \fi
      \fi
      \ifcsname\csde!PN\endcsname\else\@nameauth@FirstFormattrue\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
      \ifcsname\csde!PN\endcsname\else\csgdef{\csde!PN}{}\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
          \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\Space{,\space}%
      \if@nameauth@NBSP\protected@edef\Space{,\nobreakspace}\fi
    \fi
    \ifx\teste\@empty
      \let\FNN\argc%
    \else
      \let\FNN\arge%
    \fi
    \ifcsname\cscd!PN\endcsname\else\@nameauth@FirstFormattrue\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
    \ifcsname\cscd!PN\endcsname\else\csgdef{\cscd!PN}{}\fi
  \fi
```

Reset all the "per name" Boolean values.

```
  \@nameauth@Lockfalse%
  \@nameauth@InAKAfalse%
  \@nameauth@FirstFormatfalse%
  \@nameauth@NBSPfalse%
  \@nameauth@AltAKAfalse%
  \@nameauth@DoCapsfalse%
  \@nameauth@Accentfalse%
  \@nameauth@AllThisfalse%
  \@nameauth@ShowCommafalse%
  \@nameauth@NoCommafalse%
  \@nameauth@RevThisfalse%
  \@nameauth@RevThisCommafalse%
```

Close the "locked" branch.

```
  \fi
```

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}
```

Name Info Database: "Text Tags"

\NameAddInfo This creates a control sequence and information associated with a given name, similar to an index tag, but usable in the body text.

```
\newcommandx\NameAddInfo[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 \NameInfo: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \NameInfo: Essential name malformed}%
    \fi
  \fi
  \ifx\testa\@empty
    \ifx\testc\@empty
      \csgdef{\csb!DB}{#4}%
    \else
      \csgdef{\csbc!DB}{#4}%
    \fi
  \else
    \csgdef{\csab!DB}{#4}%
  \fi
}
```

\NameQueryInfo This prints the information created by \NameAddInfo if it exists.

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

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

```
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \NameInfo: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \NameInfo: Essential name malformed}%
    \fi
  \fi
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
    \else
      \ifcsname\csbc!DB\endcsname\csname\csbc!DB\endcsname\fi
    \fi
  \else
    \ifcsname\csab!DB\endcsname\csname\csab!DB\endcsname\fi
  \fi
}
```

`\NameClearInfo` This deletes a text tag. It has the same structure as `\UntagName`.

```
\newcommandx*\NameClearInfo[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!DB}%
    \else
      \global\csundef{\csbc!DB}%
    \fi
  \else
    \global\csundef{\csab!DB}%
  \fi
}
```

Index Operations

\IndexName This creates an index entry that is not already a pseudonym. It prints nothing. It does ensure consistent formatting. First we make copies of the arguments to test them and make parsing decisions.

```
\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}}%
```

Test for malformed input.

```
  \ifx\testb\@empty
    \PackageError{nameauth}%
    {macro \IndexName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \IndexName: Essential name malformed}%
    \fi
  \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%
    \let\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 for a "pseudonym" control sequence (code !PN) created 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
}
```

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

```
\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 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
}
```

\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
}
```

Name Decisions

`\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 \IfFrontName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \IfFrontName: 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 \IfMainName: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \IfMainName: 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
}
```

`\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}}%
  \def\test{!}%
```

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 \IfAKA: Essential name missing}%
  \else
    \ifx\csb\@empty
      \PackageError{nameauth}%
      {macro \IfAKA: Essential name malformed}%
    \fi
  \fi
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!PN\endcsname
        \edef\testa{\csname\csb!PN\endcsname}%
        \ifx\testa\test{#6}\else{#4}\fi
      \else{#5}\fi
    \else
      \ifcsname\csbc!PN\endcsname
        \edef\testa{\csname\csbc!PN\endcsname}%
        \ifx\testa\test{#6}\else{#4}\fi
      \else{#5}\fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \edef\testa{\csname\csab!PN\endcsname}%
      \ifx\testa\test{#6}\else{#4}\fi
    \else{#5}\fi
  \fi
}
```

Changing Name Decisions

\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@MainFormat selects the type of name.

```
  \ifx\testa\@empty
    \ifx\testc\@empty
      \if@nameauth@LocalNames
        \if@nameauth@MainFormat
          \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@MainFormat
          \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@MainFormat
        \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@MainFormat selects the type of name.

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

```
  \else
    \if@nameauth@LocalNames
      \if@nameauth@MainFormat
        \csgdef{\csab!MN}{}%
      \else
        \csgdef{\csab!NF}{}%
      \fi
    \else
      \csgdef{\csab!MN}{}%
      \csgdef{\csab!NF}{}%
    \fi
  \fi
}
```

Simplified Interface

`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}}%
    \@nameauth@etoksb\expandafter{##2}%
    \@nameauth@etoksc\expandafter{##3}%
    \@nameauth@etoksd\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\@nameauth@etoksc}}%
            \ex\csgdef\ex{\ex L\ex\@arga\ex}\ex{%
              \ex\@nameauth@FullNametrue%
              \ex\NameauthLName\ex{\the\@nameauth@etoksc}}%
            \ex\csgdef\ex{\ex S\ex\@arga\ex}\ex{%
              \ex\@nameauth@FirstNametrue%
              \ex\NameauthFName\ex{\the\@nameauth@etoksc}}%
```

```
          \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\@nameauth@etoksb\ex]\ex{\the\@nameauth@etoksc}}%
            \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\@nameauth@etoksb\ex]\ex{\the\@nameauth@etoksc}}%
            \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\@nameauth@etoksb\ex]\ex{\the\@nameauth@etoksc}}%
          \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\@nameauth@etoksc\ex}\ex[\the\@nameauth@etoksd]}%
            \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\@nameauth@etoksc\ex}\ex[\the\@nameauth@etoksd]}%
            \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\@nameauth@etoksc\ex}\ex[\the\@nameauth@etoksd]}%
          \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\ex\ex\NameauthName%
              \ex\ex\ex\ex\ex\ex\ex\ex[\ex\ex\ex\the\ex\ex\ex\ex\@nameauth@etoksb%
              \ex\ex\ex]\ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
              \the\@nameauth@etoksd]}%
            \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
              \ex\ex\ex\ex\ex\ex\ex\ex L\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\ex\ex\ex\@nameauth@FullNametrue%
              \ex\ex\ex\ex\ex\ex\ex\ex\NameauthLName\ex\ex\ex\ex\ex\ex\ex\ex[%
              \ex\ex\ex\the\ex\ex\ex\ex\@nameauth@etoksb%
              \ex\ex\ex]\ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
              \the\@nameauth@etoksd]}%
            \ex\ex\ex\ex\ex\ex\ex\csgdef\ex\ex\ex\ex\ex\ex\ex{%
              \ex\ex\ex\ex\ex\ex\ex\ex S\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\ex\ex\ex\@nameauth@FirstNametrue%
              \ex\ex\ex\ex\ex\ex\ex\ex\NameauthFName\ex\ex\ex\ex\ex\ex\ex\ex[%
              \ex\ex\ex\the\ex\ex\ex\ex\@nameauth@etoksb\ex\ex\ex]%
              \ex\ex\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
              \the\@nameauth@etoksd]}%
          \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.