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

Charles P. Schaum†

Released 2016/03/18

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

1 Introduction **2**
1.1 Preliminaries 2
1.2 What's In A Name? 3

2 Usage **5**
2.1 Package Options 5
2.2 Quick Start Guide 8
2.2.1 Main Interface 8
2.2.2 Simplefied Interface . 10
2.2.3 Older Syntax 12
2.3 Naming Macros 13
2.3.1 \Name and \Name* . . 13
2.3.2 Forenames: \FName . . 14
2.4 Affixes and Eastern Names . 15
2.4.1 Affixes Need Commas 15
2.4.2 Eastern Names 16
2.5 Other Naming Topics 17
2.5.1 Particles 17
2.5.2 Formatting Initials . . 18
2.5.3 Hyphenation 18
2.5.4 Listing by Surname . 19
2.5.5 Fault Tolerance 19
2.5.6 Detecting Punctuation 19
2.5.7 Accented Names . . . 20
2.5.8 Custom Formatting . 20
2.5.9 Disable Formatting . . 22
2.6 Name Decisions 23
2.6.1 Testing Decisions . . . 23
2.6.2 Changing Decisions . . 25
2.7 "Text Tags" 25
2.8 Name Variant Macros 26
2.8.1 \AKA 26
2.8.2 \PName 29
2.9 Indexing Macros 30
2.9.1 Indexing Control . . . 30
2.9.2 Indexing and babel . . 30
2.9.3 \IndexName 30
2.9.4 Index Sorting 31
2.9.5 \TagName 31
2.9.6 \UntagName 32
2.9.7 Global Name Exclusion 33
2.10 Longer Examples 34
2.10.1 Tips for \AKA 34
2.10.2 Unicode and NFSS . . 35
2.10.3 LaTeX Engines 37
2.10.4 \LocalNames 38
2.10.5 Formatting Hooks . . 39
2.10.6 Variant Spellings . . . 45
2.11 Naming Pattern Reference . . 46
2.11.1 Basic Naming 46
2.11.2 Particles 49
2.12 Errors and Warnings 50

3 Implementation **52**
3.1 Boolean Values 52
3.2 Hooks 53
3.3 Package Options 54
3.4 Internal Macros 55
3.5 User Interface Macros 67

4 Change History **91**

5 Index **93**

*This file describes version v2.42, last revised 2016/03/18.

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

1 Introduction

1.1 Preliminaries

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

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

1. Format and vary name forms according to standard syntax in the body text, independent of the index.
    - Default to long name references first, then shorter ones.
    - Use alternate names only in the body text, not the index.
    - Perform name caps and reversing only in the body text.
2. Perform typographic formatting of names only in the body text. Reflect source text typography with English conventions, but only *after* syntactic formatting is complete.
3. Allow typographic formatting to be customized and permit control sequences in names (Sections 2.5.7, 2.5.8, and 2.9.4) to allow Continental and non-English standards.
4. Always aim to reduce keystrokes.
5. Accommodate the broadest set of names while minimizing keystrokes.

This manual performs something of a "torture test" on this package. You might want to avoid doing that if you are a beginner. It is designed to be compatible with A4 and US letter. Macro references are minimized for a "clean" index, showing how nameauth handles indexing.

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

Indexing generally conforms to the standard in Nancy C. Mulvany, *Indexing Books* (Chicago: University of Chicago Press, 1994). This should be suitable for a very wide application across a number of disciplines.

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

This documentation uses names of living and historical figures because users refer to real people. At no time do I intend any disrespect or statement of bias regarding any particular person, culture, or tradition. All names are used only for teaching purposes.

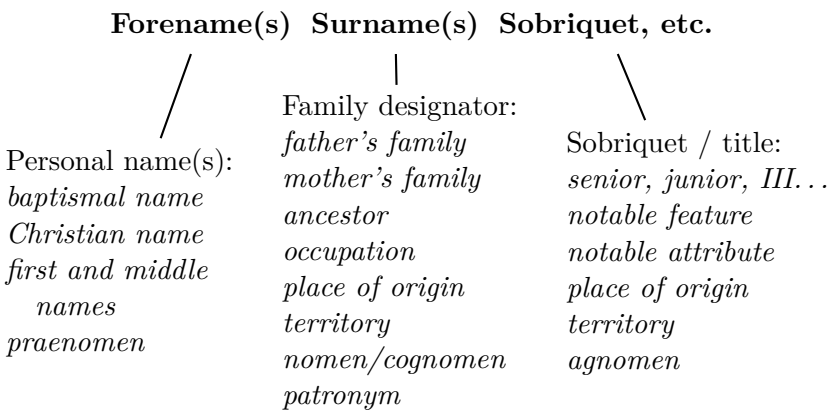
1.2 What's In A Name?

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

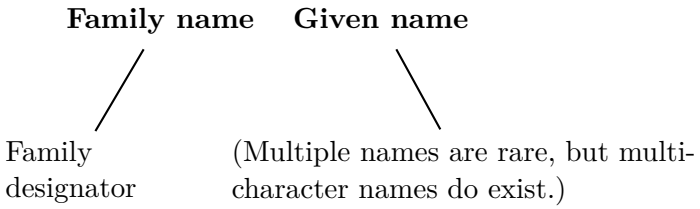
In this manual we refer to three classes of names. A "surnames" argument, ⟨*SNN*⟩, denotes a "required name," that is, a Western surname, an Eastern family name, or an ancient/medieval name.[1] Other naming systems can be adapted to these categories, *e.g.*, Icelandic, Hungarian, etc.

Professional writing often calls for the full form of a person's name to be used in its first occurrence, with shorter forms used thereafter. This package adapts that principle to all the forms below.

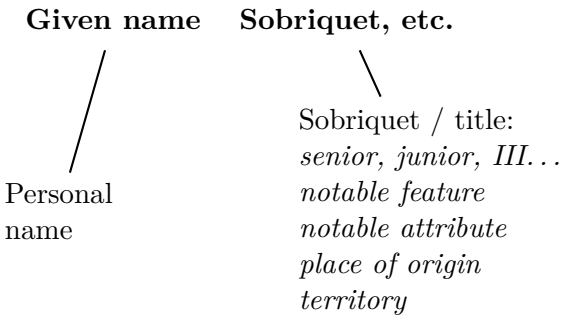
1. Western name:



2. Eastern name:



3. Ancient name:



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

Another way to think about these over-generalized classes of names is to pretend that you know *nothing at all* about names. How would you make sense out of the following?

Longer Name	Shorter Name	Indexed Name
George Washington	Washington	Washington, George
John David Rockefeller II	Rockefeller	Rockefeller, John David, II
Clive Staples Lewis	C.S. Lewis	Lewis, Clive Staples
Clive Staples Lewis	Jack Lewis	Lewis, Clive Staples
Yamamoto Isoroku	Yamamoto	Yammamoto Isoroku
Aristotle	Aristotle	Aristotle
Elizabeth I	Elizabeth	Elizabeth I
Attila the Hun	Attila	Attila the Hun

The position of a name alone does not reveal whether it is a personal or family name. You have to know the cultures of the respective names, what the affixes mean, what triggers the different handling of affixes, and so on.

Some of the humanities literature says that you should view the forenames as essential and the surnames as optional. Yet problems emerge when you have to drop the forename reference in professional or courteous writing, when you have to consider multiple alternate names, and when you have to consider those cases where the family name comes first.

Nevertheless, if you notice a few consistent relationships, that will help you work out how the nameauth package encodes names:

1. The long form of a name corresponds with the reordered indexed form.
2. All Western index forms have surname(s) first, followed by a comma, then the forename(s), then a comma if needed, then the affix if it exists.
3. All Western name forms have both forename(s) and surname(s). The forename(s) can change/drop in the text, but not in the index.
4. Eastern name forms ⟨*Surname(s)*⟩ ⟨*Forename(s)*⟩, with royal and ancient name forms ⟨*Name(s)*⟩ ⟨*Affix*⟩, should not have commas in their index entries. The forename(s) and affixes cannot change (in this package), but they can drop in the text, but not the index.

From this we conclude that all naming macros must be given the long name information in their arguments, then decide what to drop, change, and reorder depending on form and context. Since Eastern, ancient, and royal names share a set of similarities, they can allow context to disambiguate their use, over against Western names, which have to be handled differently.

All this being said, the publisher's way of handling things may trump the (arguably) canonical way. This package allows some "borderline" options and name forms to accommodate that.[2] The most prominent example includes the "non-native" Eastern names discussed in Sections 2.1 and 2.4.2.

[2]Some publishers not only use commas with Eastern forms in the index, but sometimes the forms are just wrong. For example, one sees Sun Yat-sen indexed as "Yat-sen, Sun" (instead of either "Sun, Yat-sen" or "Sun Yat-sen") in Immanuel Geiss, *Personen: Die biographische Dimension der Weltgeschichte*, Geschichte Griffbereit vol. 2 (Munich: Wissen Media Verlag, 2002), 720. The six-volume series is otherwise a helpful resource.

2 Usage

2.1 Package Options

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

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

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

Default options are in boldface.

Enable/Disable Features

Enable/Disable Formatting

`mainmatter`	**Choose "main-matter names" and formatting hooks (see formatting options below), starting at the beginning of a document.**
`frontmatter`	Choose "front-matter names" and hooks; retain syntactic formatting.

The default `mainmatter` option starts formatting names immediately. Use the `frontmatter` option to suppress name formatting until you want it to start via `\NamesActive`. These options have no additional effects on the index, but they implement two completely separate systems of first/subsequent names. These systems manage names in separate contexts. See Section 2.5.9.

Enable/Disable Indexing

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

The default `index` option enables name indexing right away. The `noindex` option disables the indexing of names until `\IndexActive` enables it. That can affect the use of index tags. This applies only to naming and indexing macros in the `nameauth` package. See Section 2.9.1.

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 in `makeindex` / `texindy`. Seldom would one change this option. 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* or Eastern names, choose the default `nocomma` option to get, *e.g.*, James Earl Carter Jr.

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

1. The `comma` option produces, *e.g.*, James Earl Carter, Jr. Yet it limits the use of macros like `\AKA` and `\PName` and it prevents the use of Eastern and ancient names with the new syntax.[3]
2. Section 2.4.1 shows how one can use `\ShowComma` with the `nocomma` option to get similar results, but with more flexibility.

Capitalize Entire Surnames

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

This only affects names printed in the body text. One of the design principles of this package keeps it consistent with English typography and syntax. Thus no syntactic or typographic changes are propagated into the index by default.

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

Section 2.4.2 deals with capitalization on a section-level and per-name basis.

Reverse Name Order

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

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

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

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

Typographic Display of Names

Section 2.5.8 explains in greater detail that typographic display is different from the syntactic formatting of names and occurs after syntactic formatting is complete. This package is designed with type hierarchies in mind.[4]

As of version 2.4, "typographic formatting" has become a generalized concept of "name post-processing" via the hook macros `\NamesFormat`, `\MainNameHook`, and `\FrontNameHook`. Sections 2.5.8 and 2.10.5 offer substantially more complex possibilities for such hooks.[5]

Even though English typography was the design choice for this package, one can use this package in Continental (German, etc.) and other typographic standards, as Section 2.5.8 discusses. One must use sort tags, as Section 2.9.4 explains.

Continental standards include formatting surnames only. However, nameauth uses some ambiguous name forms.[6] Continental users must add control sequences directly in the naming macro arguments. In that case, use the `noformat` option and the simplified interface to minimize keystrokes. Continental users may lose some capitalization features; see Section 2.5.1.

Otherwise, the options below are meant generally for applications in English typography. The default is `smallcaps` because this package was developed to aid my editing and translation of older German and Latin documents into English. I do apologize for any inconvenience in design choices.

Formatting Attributes

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

[4] See Robert Bringhurst, *The Elements of Typographic Style*, version 3.2 (Point Roberts, Washington: Hartley & Marks, 2008), 53–60.

[5] 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.

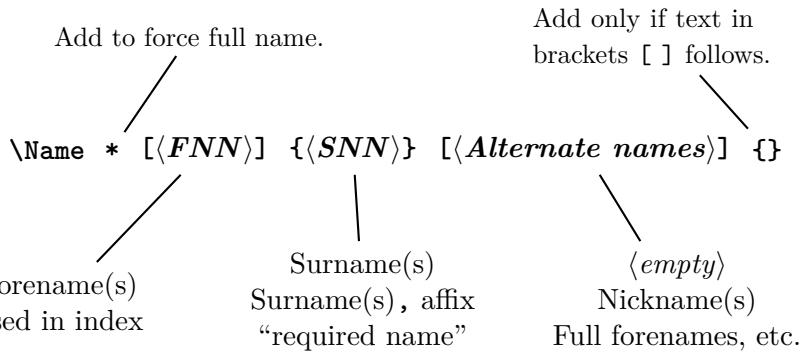
[6] By default, nameauth uses a "serendipitous ambiguity" of ancient, Eastern, and suffixed name forms handled by the ⟨*Surname, affix*⟩ pattern that is resolved subtly by several factors.

2.2 Quick Start Guide

2.2.1 Main Interface

See Section 2.3 for a proper description of `\Name`. Here we see briefly how to work with the classes of names in Section 1.2. We abbreviate the macro arguments ⟨*forename(s)*⟩ with ⟨*FNN*⟩ and ⟨*surname(s)*⟩ with ⟨*SNN*⟩. Use the `nocomma` option especially when using Eastern names and ancient names.

Western Names



Usual forms:

`\Name[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}` `\Name[George]{Washington}`

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

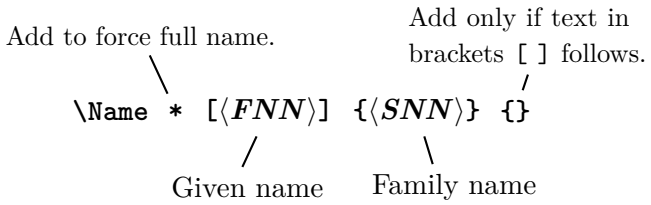
You must include the ⟨*FNN*⟩ field with alternate forenames. The ⟨*Alternate names*⟩ are swapped with the ⟨*FNN*⟩, but only in the body text:

`\Name[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[`⟨*Alternate names*⟩`]`
`\Name[Bob]{Hope}[Leslie Townes]`
`\Name[Clive Staples]{Lewis}[C.S.]`

`\Name[`⟨*FNN*⟩`]{`⟨*SNN, affix*⟩`}[`⟨*Alternate names*⟩`]`
`\Name[John David]{Rockefeller, IV}[Jay]`

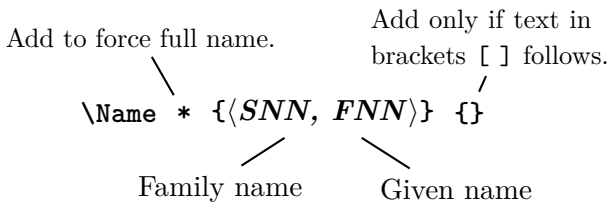
The older syntax is `\Name{`⟨*SNN*⟩`}[`⟨*affix*⟩`]`. See Section 2.2.3 for its usage and its shortcomings. It remains for backward compatibility.

Eastern Names in the Text, Western-style Index



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

Eastern Names in the Text, Eastern-style Index



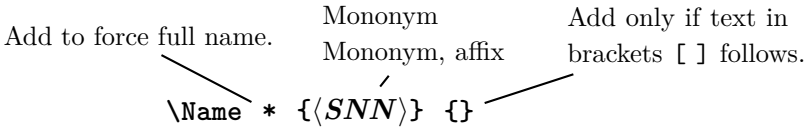
Usual form:

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

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

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

Ancient Names



Usual form:

`\Name{`⟨*SNN*⟩`}` `\Name{Aristotle}`

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

`\Name{Attila, the Hun}`

These forms are used for royalty, ancient figures, and other monomyms with or without suffixes.[7] The older syntax takes the form `\Name{`⟨*Mononym*⟩`}[`⟨*affix*⟩`]`. Cf. Section 2.2.3.

Many commands in the main interface are variants of a base pattern, where ⟨*prefix macro*⟩ consist of one or more of `\CapThis`, `\CapName`, `\RevName`, `\RevComma`, `\ShowComma`, and `\KeepAffix`:

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

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

2.2.2 Simplefied Interface

nameauth

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

The simplified interface produces control sequences that are fully compatible with the main interface. Although not required, `nameauth` is best used in the document preamble to avoid undefined control sequences.[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*.
- Leading and trailing spaces in each `&`-delimited field are stripped, as is also the case in the main interface.
- As in the main interface, medial spaces do not affect first-use control sequences, but they will affect name forms in the body text and index.
- In the document text, as with the main interface, include trailing braces `{}`, control spaces, or the like if text in brackets `[]` follows any of the shorthands, *e.g.*, `\LWash{} [\emph{sic}]`.
- The old syntax (Section 2.2.3), triggered by an empty ⟨*FNN*⟩ field, causes the ⟨*Alt. names*⟩ field to be interpreted as either Eastern ⟨*FNN*⟩ or an ⟨*affix*⟩. Due to its limitations and potential confusion, you are encouraged to avoid it unless you are using the `comma` option.

[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.

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

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

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

Basic Uses:

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

Western Reversed with Comma:

`\RevComma\LWash`	Washington, George

Particles:

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

Affixes:

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

Nicknames: (See Section 2.3.2)

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

Ancient:

`\Aris`	Aristotle
`\Aris`	Aristotle

Medieval/Royal:

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

Western / Western Index:

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

Eastern / Western Index:

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

Eastern / Eastern Index:

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

Western / Eastern Index:

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

Sections 2.5.1, 2.5.7, and 2.9.4 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 (Section 2.9.4), the ⟨*Alternate names*⟩ field has no effect on index tags. `\JRIV` and `\JayR` need only one tag:

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

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

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

2.2.3 Older Syntax

An "obsolete" syntax remains for backward compatibility with early versions of `nameauth` and with the `comma` option. Please avoid mixing the older and newer forms to avoid possible confusion and error. For example, the older syntax causes the ⟨*Alternate names*⟩ field in the index tagging functions to become as significant as ⟨*FNN*⟩, including the need to pretag such names.

The `comma` option causes Western names with affixes to have a comma. Yet that also causes Eastern and ancient names, or any names using a pattern like ⟨*SNN, affix*⟩ or ⟨*SNN, FNN*⟩ to display a comma where it should not occur. The old form lacks some error checking and robustness contained in the new syntax and limits the use of several macros, including `\AKA`. Section 2.12 offers some cautions about the old syntax, as do many places in this manual. The form is:

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

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

`\Dagb` gives Dagobert I, then Dagobert. In similar fashion, we see `\LDagb` Dagobert I, `\CapName\Yosh` YOSHIDA Shigeru, and `\CapName\RevName\LYosh` Shigeru YOSHIDA.

In the old syntax, `\Name{Henry}[VIII]` prints "Henry VIII" and "Henry." If you mix `\Name{Henry}[VIII]` with the newer `\Name{Henry, VIII}` they both print Henry VIII and Henry in the body text. Yet they generate different control sequences for both first/subsequent uses and index tags.[13]

Avoid `\Name{Henry, VIII}[Tudor]` unless you want "Henry VIII Tudor" and "Henry" in the body text and "Henry VIII Tudor" in the index. One solution adds "Tudor" as needed in the text after `\Name{Henry, VIII}` and uses a tag in the index: `\TagName{Henry, VIII}{ Tudor}` (see Section 2.9.5).

[13] Technically you can mix the two, as I do here. You must force first and subsequent uses with `\ForgetName` and `\SubvertName`. You must make common index tags, *e.g.*: `\TagName{Henry, VIII}{, king}` and `\TagName{Henry}[VIII]{, king}`. That undermines the time-saving features offered by this package.

2.3 Naming Macros

2.3.1 `\Name` and `\Name*`

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

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

Here we see how the syntax works:

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

`\Name` displays and indexes names, as illustrated in Section 2.11. 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 and Confucius. `\Cicero` prints M.T. Cicero and Cicero, while `\LCicero[Marcus Tullius]` gives Marcus Tullius Cicero. The next page explains why this form may be preferable in some cases for name variants when using the simplified interface.

2.3.2 Forenames: \FName

\FName
\FName*

\FName and its synonym \FName* print just forenames, but only in subsequent name uses.[14] They are intended for Western-style names. The syntax is:

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

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

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

By design, \FName *never* prints Eastern personal names, so that ancient names also work (cf. Section 2.11). 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.

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

2.4 Affixes and Eastern Names

2.4.1 Affixes Need Commas

Comma-delimited affixes handle several different name types. *Always include a comma as an affix delimiter*, even when the `nocomma` option does not print the comma. Extra spaces between the comma and affix are ignored. Extra commas have no effect. Other name types include royal, medieval, and Eastern names:

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

You cannot use the old syntax with the Hammerstein example. One must use comma-delimited suffixes when cross-referencing affixed Western names, royal names, some medieval names, and Eastern names with `\AKA`; see Section 2.8.1.

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

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

Compare Older Syntax

Avoid using the older syntax, shown below, except with the `comma` option. The older syntax causes Eastern and ancient names that use the ⟨*SNN, affix*⟩ pattern to have unwanted commas in them with the `comma` option or with `\ShowComma`. `\AKA` and `\PName` cannot create cross-references to these forms:

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

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

2.4.2 Eastern Names

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

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

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

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

`\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.

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

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

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

Use `\AllCapsActive`, `\AllCapsInactive`, and `\CapName` for fully-capitalized family names in the body text. These macros are analogous to the reversing macros above and may be used alone or with those and other state-toggling macros, *e.g.* `\CapName\RevName\Name`....

`\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. Forms using the old syntax are in parentheses. Name formatting is turned off in order to focus on capitalizing and reversing:

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

Notice how capitalization is independent of formatting. The reversing and capitalization macros also work with `\AKA`. They affect only the text, not the index. For caps in the text and index see Section 2.5.8.

2.5 Other Naming Topics

Language-Related Issues

2.5.1 Particles

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

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

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 2.10.2).

The Continental style of handling surnames (Section 2.5.8) does not work with the capitalization macros. In the case of 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}}`.

`\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 2.10.2). 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 *Ĺ* and *d́* 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`.

Another example deals with particles and name forms:

```
\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.[15]

Many people use medieval affixes as Western surnames: "À Kempis."[16] To get that form, use `\AccentCapThis\KempW`.

[15] 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.

[16] This treatment of medieval names, along with the handling of Eastern names, seems to be one of the most frequently "abused" issues in the academic literature that I know. In order to achieve simplicity in work flow or conformity, authors and publishers will take some fairly ethnocentric or heavy-handed approaches to names. The `nameauth` package will accommodate those approaches, even if I personally disagree with them.

Section 2.10.2 explains in detail why the following problems can occur:

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

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 2.10.2 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.

2.5.2 Formatting Initials

Omit spaces between initials if possible; see also Bringhurst's *Elements of Typographic Style*. If your publisher wants spaces between initials, try putting thin spaces `\,` between them. 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 and E. B. White
Normal text:	E. B. White and E. B. White.

2.5.3 Hyphenation

The simplified interface trivializes the insertion of optional hyphens in names, but such hyphens must be used consistently in all the naming macros:

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

We get Johann Biermann and Biermann. English hyphenation can break ie-pairs and maybe others. One also can fix bad breaks with the babel or polyglossia packages. This moves the solution from "quick and dirty" to elegant. John Strietelmeier can break badly in English, as you see. Using babel, we get:

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

2.5.4 Listing by Surname

\ReverseCommaActive
\ReverseCommaInactive
\RevComma
\global

The reversing macros \ReverseCommaActive, \ReverseCommaInactive, and \RevComma allow the easy generation of name lists ordered as ⟨*SNN*⟩, ⟨*FNN*⟩ or ⟨*SNN*⟩, ⟨*Alt. names*⟩. The first two are broad toggles, while the third works on a per-name basis. 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. Name formatting has been turned off to focus on reversing and commas:

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

Technical-Related Issues

2.5.5 Fault Tolerance

Especially since version 2.0, the nameauth package minimizes possible side effects of malformed input and macros that expand to being empty in required values. To reduce errors, \Name, \FName, \AKA, and \IndexName ignore leading and trailing spaces — but not medial spaces — making the following equivalent:

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

2.5.6 Detecting Punctuation

In Western names, affixes like "Jr." (junior), "Sr." (senior), "d. J." (*der Jüngere*), and "d. Ä." (*der Ältere*) can colide with the full stop in a sentance. \Name, \FName, and \AKA check for a trailing period in the name that they print in the text. If they find it, they check if the next token is a full stop and gobble it if so:

Macro Example	*periods*	*Resulting Text*
`\Name[Martin Luther]{King, Jr.}.`	$2 \to 1$	Martin Luther King Jr.
`\Name[Martin Luther]{King, Jr.}.`	$2 \to 1$	King.
`\Name[Martin Luther]{King, Jr.}␣`	$1 \to 0$	King
`\Name*[Martin Luther]{King, Jr.}.`	$2 \to 1$	Martin Luther King Jr.
`\Name*[Martin Luther]{King, Jr.}␣`	$1 \to 1$	Martin Luther King Jr.
`{\Name*[Martin Luther]{King, Jr.}}.`	$2 \to 2$	Martin Luther King Jr..[17]

[17] Grouping tokens and other items can frustrate the full stop detection mechanism.

2.5.7 Accented Names

For names that contain accented characters, using `xelatex` or `lualatex` with `xindy` (`texindy`) is recommended. Section 2.10.3 shows how you can work with multiple engines.

If the leading character of ⟨*SNN*⟩ is accented and lowercase (usually only in a particle), then you must use `\AccentCapThis` if you are using `pdflatex`. Sections 2.5.1 and 2.10.2 give more details about `\CapThis` and `\AccentCapThis`.

Accented characters act like control sequences. In `pdflatex` use `\PretagName` with all names with extended Unicode characters (Sections 2.9.4 and 2.10.2).[18]

Nevertheless, Unicode characters and "regular" control sequences are not interchangeable. The example below shows this difference because the names are all long (thus, different). The names are not long, then short (were they the same):

`\Name[Johann]{Andre\"a}`	JOHANN ANDREÄ
`\Name[Johann]{Andreä}`	JOHANN ANDREÄ instead of Andreä
`\Name{\AE thelred, II}`	ÆTHELRED II
`\Name{Æthelred, II}`	ÆTHELRED II instead of Æthelred

See Section 2.10.2 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 define letters with `\edef` and `\noexpand` to use in names, as some do to "protect" accented letters in names. As of version 2.0 of nameauth helpful concerns expressed by PATRICK COUSOT have been addressed.

2.5.8 Custom Formatting

There are two kinds of formatting at work:

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

Continental small caps

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

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

You get Greta GARBO, then GARBO, even in the front matter because this formatting is persistent. Use `‘‘Garbo’’\IndexName\Name[Greta]{\textsc{Garbo}}` for a "Garbo" reference. `\Name[\normalfont{Greta}]{\textsc{Garbo}}` may look like the name above, but it is a different name with a different index entry. Keep the formatting simple to gain both flexibility and consistency.

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

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

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

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

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

`\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.

`\NamesFormat`

When formatting is active, `\NamesFormat` is called at the first instance of a name, and at every instance of a name when the `alwaysformat` option is used. Originally it was the only formatting "hook," but in version 2.4 and beyond it joins two similar "hooks" that are described in Section 2.10.5. One can redefine `\NamesFormat` to create custom effects like suppressing formatting in footnotes:

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

Your footnote would produce an unformatted name.[19] We change footnotes back to normal with `\makeatletter\let\@makefntext\@oldfntext\makeatother`. Yet this example also can affect names in the body text, which may not be desirable. Section 2.5.9 has an arguably better approach.

Relying only on scoping to insulate changes to `\NamesFormat` might create undesired side effects, depending on the sort of modification. We put longer examples of formatting and conditionals in Section 2.10.5.

2.5.9 Disable Formatting

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

`\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.

Here we switch to the "front matter" mode by placing `\NamesInactive` at the start of the `quote` environment:

```
\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" by leaving the scope above instead of calling `\NamesActive` explicitly. Avoid unwanted effects by using these toggling macros in pairs (perhaps with `\global`).

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

Notice that we have two independent cases of "first use" above. Consider the two "species" of names to be "non-formatted" and "formatted," intended for front matter and main matter. Yet one could use this in footnotes to implement a different system of names (see also Section 2.6.2):

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

As above, your footnote would produce an unformatted name.[20] Here the "non-formatted" names would not affect the body text. Again we change footnotes back to normal with `\makeatletter\let\@makefntext\@oldfntext\makeatother`.

[19] This demonstrates no main-matter formatting: John Maynard Keynes.
[20] This demonstrates no front-matter formatting: John Maynard Keynes.

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. The `\If` in this section's macro names is capitalized to be different from a regular `\if` expression. The branching of these macros is altered by using `\Name`, `\Name*`, `\FName`, `\PName`, `\AKA`, `\AKA*`, `\ForgetName`, `\SubvertName`, and `\ExcludeName`.

Some examples of conditional text include a "mini-biography," a footnote, or a callout. These macros could be integrated with the "text tag" features in Section 2.7. Authors and editors could use these macros with the comment, pdfcomment, and similar packages to make comments based on whether a name has occurred or not. That aids name management and thought development.

`\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 Hope" from the following example because we have yet to invoke `\Name[Bob]{Hope}`:

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

`\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}}
```

We will return to this topic of main matter and front matter names later in Sections 2.6.2 and 2.10.4. There we see how `\ForgetName` and `\SubvertName` usually affect both main- and front-matter names simultaneously unless set otherwise.

`\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.

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 23 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.[21]

`\LocalNames` If, however, this "global" behavior of `\ForgetName` and `\SubvertName` is not
`\GlobalNames` 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 via the `frontmatter` option or `\NamesInactive`, `\ForgetName` and `\SubvertName` will only affect unformatted names. If you are in a "main matter" section via the `mainmatter` option or `\NamesActive`, then `\ForgetName` and `\SubvertName` will only affect formatted names. Section 2.10.4 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

[21] 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.

every name managed by nameauth. Section 2.10.5 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.

\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 2.10.5 illustrate how this can permit tags like asterisks, daggers, and footnotes in addition to tags that do need a space or some separation between them and the name.

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

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

Only the ⟨*FNN*⟩ and ⟨*SNN*⟩ arguments from \Name and friends may be cross-referenced. The new syntax allows \AKA to cross-reference all name types. Both macros create a cross-reference in the index from the ⟨*Alt. FNN*⟩, ⟨*Alt. SNN*⟩, and ⟨*Alt. names*⟩ fields to a name defined by ⟨*FNN*⟩ and ⟨*SNN*⟩, regardless of whether that name has been used. Please consult also Section 2.10.1, which covers a number of in-depth examples of \AKA.

Both macros print only the ⟨*Alt. FNN*⟩ and ⟨*Alt. SNN*⟩ fields in the body text. If the ⟨*Alt. names*⟩ field is present, `\AKA` swaps ⟨*Alt. names*⟩ with ⟨*Alt. FNN*⟩ in the body text, similar to the naming macros.

`\AKA*` has two functions. For Western names, where ⟨*Alt. FNN*⟩ is present, `\AKA*` prints either just the ⟨*Alt. FNN*⟩ or just the ⟨*Alt. names*⟩ when they also are present. However, if ⟨*Alt. FNN*⟩ is absent, `\AKA*` prints just ⟨*Alt. names*⟩ if present, otherwise ⟨*Alt. SNN*⟩. See also Section 2.9.5.

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

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

JEAN SANS PEUR (Jean the Fearless) was Duke of Burgundy 1404–1419.

"Jean the Fearless" receives no formatting; this package usually formats only names with main index entries. Section 2.10.5 discusses how one might address that beyond the `alwaysformat` option. The syntactic aspects of name formatting (caps and reversing) always work with `\AKA`.

The following complex example has lines of source literals interleaved with a point-by-point enumerated list, showing the Continental style. The small caps are a syntactic element of the name parameters themselves:

1. I tag the names for proper sorting.

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

2. I want "Heinz RÜHMANN" to be the main name of reference, so `\AKA*` uses his legal name as the cross-reference. `\AKA*` prints only "Heinrich Wilhelm" in the body text. Nevertheless, the index cross-reference will be complete with the surname.

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

3. `\SubvertName` causes `\FName` to print the short version. It also bypasses the default name formatting.

   ```
   \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}} %
   ```

The resulting text is:

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

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

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

The alternate form "Lester T. Hope" in the previous table does not appear in the index, but only in the body text. A possible use here could involve "spurious" information or opinions that one might want to mention in the text but not the index. One produces Gregory I "the Great," along with a *see* reference from "Gregory the Great" to "Gregory I," via:

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

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

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

Cross-references generated by `\AKA` and `\AKA*` are meant only to be *see* references, never page entries. See also Section 2.12. 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.[22]

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:

[22] 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.

```
\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 2.5.1 talks about those unfortunate situations where one must use the Western form `\Name[Thomas]{Aquinas}`.

2.8.2 \PName

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

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

Although `\PName` creates an easy shortcut, its drawbacks are many. It only can use the ⟨*FNN*⟩⟨*SNN*⟩ form of `\AKA`. Neither `\AKA*`, nor `\CapName`, `\CapThis`, `\RevComma`, `\RevName`, and the related package options work with `\PName`.

The main name comes first, followed by the name that is only a *see* reference. `\PName` can generate the following examples:

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

Mark Twain (Samuel L. Clemens)
Mark Twain (Samuel L. Clemens)
Twain (Samuel L. Clemens)

```
\PName{Voltaire}[François-Marie]{Arouet}
\PName*{Voltaire}[François-Marie]{Arouet}
\PName{Voltaire}[François-Marie]{Arouet}
```

Voltaire (François-Marie Arouet)
Voltaire (François-Marie Arouet)
Voltaire (François-Marie Arouet)

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

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

2.9 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.

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

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 use the following form that works with both `makeindex` and `texindy`: `\index{`⟨*sortkey*⟩`@`⟨*actual*⟩`}`

Before version 2.0 of `nameauth`, one had to sort and index names like Jan Łukasiewicz and Æthelred II by putting them between `\IndexInactive` and `\IndexActive` while creating manual index entries.

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

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

The `\PretagName` macro differs from the other tagging macros:

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

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

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

One need only pretag names once in the preamble. Every time that one refers to Łukasiewicz or Æthelred, the proper index entry will be created. If you create a cross-reference with `\AKA` and you want to pretag it, see Section 2.8.1.

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

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

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.[23]

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).

[23] 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 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`: Hope, Leslie Townes
  `\RevName`: Yakumo KOIZUMI
- The name fields of `\PretagName` correspond with the `[`⟨*Alt. FNN*⟩`]{`⟨*Alt. SNN*⟩`}[`⟨*Alt. names*⟩`]` fields of `\AKA`:

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

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

  Using `\AKA` with such names looks like: `\AKA[The Amazing]{Kreskin}[Joseph]{Kresge}` and `\AKA[J.]{Kreskin}[Joseph]{Kresge}`.You get The Amazing Kreskin, a.k.a. Joseph Kresge.
- Special cases like "Iron Mike" Tyson as the nickname for Mike Tyson may be handled in a number of ways.

  1. Follow ``Iron Mike'' with `\IndexName[Mike]{Tyson}` and do whatever you want in the text. This may be the easiest solution.
  2. Use ``\AKA[Mike]{Tyson}{Iron Mike}'' to create "Iron Mike" in the text and a *see*-type cross-reference to "Tyson, Mike" in the index. Be sure to have an occurrence of `\Name[Mike]{Tyson}` in the text. See also Section 2.8.1. This is the best solution in terms of how nameauth is designed.
  3. Always get "Iron Mike Tyson" with something like:

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

     ``\Iron'' gives you "Iron Mike Tyson."[24] 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.

[24] 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.

2.10.2 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 IJ Ľ Ŀ Ł ł	Ń ń Ň ň Œ œ	Ŕ ŕ Ř ř	SMALL CAPS
IJ ij Ľ ľ 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 ť	Ů ů Ű ű	Ź ź Ż ż Ž ž	normal

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 `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 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:[25]

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>

Sections 2.5.1 and 2.5.7 have information related to these topics and the nameauth package.

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

2.10.3 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:

```
\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 has been tested with all three engines. 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
```

2.10.4 \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 an unformatted section and create a name there. Observe that \global precedes \NamesInactive because we want those effects to persist beyond the immediate scope of the quote environment:

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

Now we are in a "front matter section." 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 local in scope. We then forget the name in the unformatted section:

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

Since the "front-matter name" was removed, only a "main-matter name" exists. We now "subvert" the front-matter name to bring its "existence" back again and switch to the main section. See that \global precedes \NamesActive because we used \global previously and want a similar effect:

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

Now both names exist again, but \ForgetName and \SubvertName are still local in scope. We forget the main-matter name and additionally reset the default behavior so that \ForgetName and \SubvertName will be global:

```
\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
```

2.10.5 Formatting Hooks

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:

```
\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 or with the old syntax, lest errors bite! You would get multiple index entries, unwanted cross-references, and unexpected forms in the text. The simplified interface greatly helps one to avoid this.

Conditionals / Text Tags

We continue onward to using not only name conditionals (Section 2.6.1) but also text tags (Section 2.7) to put tags after first references to main-matter names.

`\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.

```
\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\IfMainName\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]%
      {}%                     skip true branch to get first use
      {%
        \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]%
      }%             this form allows accents/control sequences
    \fi
  \fi
  \if@nameauth@InAKA%                             do only in \AKA
    \ifNoTextTag\else
      \ex\ex\ex\ex\ex\ex\ex\IfAKA%
      \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]%
      {}%                     skip true branch to get first use
      {%
        \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]%
      }%             this form allows accents/control sequences
      {}%                                  skip excluded branch
    \fi
  \fi
  \global\NoTextTagfalse%                reset tag suppression
}
\makeatother%
```

The example above uses `\NoTextTagtrue` to suppress tags and prints then by default in the false path. A different approach would print the tags only on the true path and still set `\NoTextTagfalse` at the end, so that the tags would only print when explicitly triggered.

Before we can refer to any text tags, we must create them. Please pardon the fact that I am going to "lie" about the tag used for "Atatürk" below in order to illustrate certain points regarding `\AKA`. I will tell the truth later when this group of examples is complete:

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

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

```
\Wash held office 1789--97. No tags appear in later uses of \Wash.
We now suppress the dates and trigger a new first use:
\NoTextTagtrue\ForgetName[George]{Washington}\Wash.

\Name[Mustafa]{Kemal} was later given the name%
\AKA[Mustafa]{Kemal}{Atatürk}.
```

George Washington (1732–99) held office 1789–97. No tags appear in later uses of Washington. We now suppress the dates and trigger a new first use: **George Washington**.

Mustafa Kemal (1881–1938) was later given the name Atatürk.

Notice that the tag for Atatürk did not print. That is because `\AKA` only prints a formatted name when `\NamesActive` is in force and the `alwaysformat` option is set. We now forget Washington and Kemal, using the same text but now simulating the effects of `alwaysformat`:

George Washington (1732–99) held office 1789–97. No tags appear in later uses of **Washington**. We now suppress the dates and trigger a new first use: **George Washington**.

Mustafa Kemal (1881–1938) was later given the name **Atatürk** (a special surname granted 1934).

Here we see that the tag is printed because `\NamesFormat` is called for every use. Still, the tags will not print with `\NamesInactive`. Before we get to the solution of that issue, perhaps it would help to think about what is happening:

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 function, which decides if it is formatted main matter, regular main matter, or front matter as the Boolean values dictate.
2. In the hooks and `\NamesFormat`:
   (a) Normally you do nothing and exit, or make a local font change and exit. You could do more complex tasks.[26]

[26] One could discard the text output of the naming macros, retaining only the indexing functionality, then re-parse the name parameters and create a custom form of text output. One possibility could allow a more flexible Continental approach with macros that expand to `\textsc` except in the hook macros, where they output format only surnames, but perhaps not always.

(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) clean up and exit.

`\MainNameHook` `\FrontNameHook`

`\AKA` can print a tag only once because its control sequences cannot be undefined by the nameauth macros. No tags are printed when `\NamesInactive` turns off formatting — even with `alwaysformat.` Two additional hooks address these limitations. `\MainNameHook` is triggered when `\NamesActive` is and `\NamesFormat` is not invoked. `\FrontNameHook` always is invoked when `\NamesInactive` is invoked. These hooks compliment `\NamesFormat.` With the examples below we use tags in the front matter and with the normal first-use case of `\AKA.`

```
\let\OldFrontHook\FrontNameHook%                  save the hook
\let\OldMainHook\MainNameHook%                    save the hook
\makeatletter%                                  access internals
\renewcommand*\MainNameHook[1]%
{%
  \let\ex\expandafter%                              reduce typing
  {#1}%
  \if@nameauth@InAKA%                             do only in \AKA
    \ifNoTextTag\else
      \ex\ex\ex\ex\ex\ex\ex\IfAKA%
      \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]%
      {}%                      skip true branch to get first use
      {%
        \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]%
      }%             this form allows accents/control sequences
      {}%                                  skip excluded branch
    \fi
  \fi
  \global\NoTextTagfalse%                reset tag suppression
}
```

```
\renewcommand*\FrontNameHook[1]%
{%
  \let\ex\expandafter%                              reduce typing
  {#1}%
  \if@nameauth@InName%               do only in \@nameauth@Name
    \ifNoTextTag\else
      \ex\ex\ex\ex\ex\ex\ex\IfFrontName%
      \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]%
      {}%                       skip true branch to get first use
      {%
        \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]%
      }%             this form allows accents/control sequences
      {}%                                   skip excluded branch
    \fi
  \fi
  \if@nameauth@InAKA%                             do only in \AKA
    \ifNoTextTag\else
      \ex\ex\ex\ex\ex\ex\ex\IfAKA%
      \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]%
      {}%                       skip true branch to get first use
      {%
        \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]%
      }%             this form allows accents/control sequences
      {}%                                   skip excluded branch
    \fi
  \fi
  \global\NoTextTagfalse%                reset tag suppression
} \makeatother%
```

We forget Washington and Kemal, then see the text as front matter:

> George Washington (1732–99) held office 1789–97. No tags appear in later uses of Washington. We now suppress the dates and trigger a new first use: George Washington.
>
> Mustafa Kemal (1881–1938) was later given the name Atatürk (a special surname granted 1934).

Finally we have the same text as main matter:

> **George Washington** (1732–99) held office 1789–97. No tags appear in later uses of Washington. We now suppress the dates and trigger a new first use: **George Washington**.
>
> **Mustafa Kemal** (1881–1938) was later given the name Atatürk (a special surname granted 1934).

Please be aware that each instance of Atatürk in the four example paragraphs above actually was a different name, even though they look the same:

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

Because the names were different, I suppressed all but one of the variants from appearing in the index by putting the undesired extras between `\IndexInactive` and `\IndexActive`.

Please do remember that `\IfAKA` will only expand to the false route before the first appearance of a pseudonym. Otherwise it will remain true for the rest of the document. Please do not let these examples mislead you in that regard. They are merely illustrative of hypothetical first uses.

The next example shows how you cannot re-enter `\Name` or `\AKA` from within `\Namesformat`, `\FrontNameHook`, or `\MainNameHook`.

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

Calling `\Wash` produces Washington. `\Name` and `\AKA` both exit and do nothing. Version 2.4 of `nameauth` prevents stack-overflows for the above example and the odd case of calling the naming macros as their own arguments. Nevertheless, `\Name{foo\Name{bar}}` would produce "FOO" in the text and "fooBAR" in the index. One might want to avoid such cases.

2.10.6 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.

2.11 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

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

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

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

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

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

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

`\Henry, \LHenry, \Henry, and \SHenry`:
Henry VIII, Henry VIII, Henry, and Henry
`\Ptol, \LPtol, \Ptol, and \SPtol`:
Ptolemy I Soter, Ptolemy I Soter, Ptolemy, and Ptolemy
`\Mao, \LMao, \Mao, and \SMao`:
Mao Tse-tung, Mao Tse-tung, Mao, and Mao

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

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

2.11.2 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. It may act as malformed input if you mix it with proper medieval name forms, but it will not affect them adversely.
- The older syntax will not work with some macros. From the film *Men in Black III*, `\AKA{Boris}[the Animal]{Just Boris}` fails. `\PName` fails for the same reasons. See also Section 2.8.1
- 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 a cross-reference `[`⟨*Alternate names*⟩`]{`⟨*Alternate SNN*⟩`}[`⟨*Alt. names*⟩`]` created by `\AKA` as a name reference in `\Name`, `\FName`, and `\PName`. They merely will print a name in the body text.
- Using a name reference `[`⟨*FNN*⟩`]{`⟨*SNN*⟩`}[`⟨*Alternate names*⟩`]` created by `\Name`, `\FName`, and `\PName` as a cross-reference in `\AKA`. It merely will print a name in the body text.
- Using `\AKA` to create the same cross-reference multiple times or with a cross-reference created by `\ExcludeName`. It merely will print a name in the body text, but not the index.
- Using `\IndexName` to index a cross-reference made via `\AKA` or via the mechanism in `\ExcludeName` as a main entry. It will do nothing.

- Using `\TagName`, `\UntagName`, and `\PretagName` with cross-references. The first two will do nothing. However, `\PretagName` will "pretag" a cross-reference. This is the desired behavior.
- Using `\ExcludeName` with cross-references. It will do nothing.
- Using `\ExcludeName` to exclude a name that has already been used. Likewise, it will do nothing.
- Using `\Name`, `\FName`, `\PName`, and `\AKA` to refer to names and cross-references excluded by `\ExcludeName`. They merely will print a name in the body text.
- Using the `nameauth` environment to redefine shorthands, such as:

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

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

  Such redefinitions could generate unwanted index entries.

3 Implementation

3.1 Boolean Values

Affix Commas

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

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

Toggle Formatting

`\NamesActive` and `\NamesInactive` or the `mainmatter` and `frontmatter` options set or clear the value below. `\@nameauth@DoFormatfalse` ensures that `\FrontNameHook` is called for all front-matter names.

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

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

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

Indexing

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

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

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

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

Syntactic Formatting

`\@nameauth@FullName` toggles long or short forms in subsequent name uses. As a corollary, `\@nameauth@FirstName` is used when printing only first names. `\@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
```

Typographic Formatting

`\@nameauth@FirstFormat` toggles the formatting of main-matter names by triggering whether `\NamesFormat` is called or whether `\MainNameHook` is called. Additionally, `\@nameauth@AlwaysFormat` forces name formatting whenever `\@nameauth@DoFormat` is true by setting `\@nameauth@FirstFormattrue`.

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

Who Called Me?

These values are true within `\Name` and `\AKA`, respectively. Otherwise they are false. Normally they have no effect, but they are available for special customizations of `\NamesFormat`, `\MainNameHook`, and `\FrontNameHook`. See Section 2.10.5.

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

As a side note, `\AKA` will invoke `\NamesFormat` if the `alwaysformat` option is set and `\NamesActive`. Otherwise it will invoke `\MainNameHook` in the main matter and `\FrontNameHook` in the front matter (`\NamesInactive`). Thus you need to check both Boolean values above to get the desired outcomes for these hooks.

Stack Overflow Prevention

Here is the locking mechanism that prevents a stack overflow via recursive calls to `\Name` and `\AKA`. See Section 2.10.5.

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

3.2 Hooks

`\NamesFormat` Change typographic formatting of final complete name form in text. See Sections 2.5.8 and 2.10.5. Called when `\@nameauth@DoFormattrue` and `\@nameauth@FirstFormattrue`.

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

`\MainNameHook` Hook for when a non-formatted main-matter name is printed (Section 2.10.5).

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

`\FrontNameHook` Hook for when a non-formatted front-matter name is printed (Section 2.10.5).

```
\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 2.5.8. 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 2.5.8.

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

`\NameauthFName` Customization hook called after `\@nameauth@FirstName` is set true. See Section 2.5.8.

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

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

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

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.[27] 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
}
```

[27] 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 following macros format the output of `\Name`, etc. `\@nameauth@FmtName` prints names in the body text, either formatted or not. Notice how `\NamesFormat` (Section 2.5.8) sits between a `\bgroup` and an `\egroup` to localize the font change. `@nameauth@AlwaysFormat` will force formatting when possible. `\@nameauth@InHook` prevents one from calling either `\Name` or `\AKA` from within the hook macros.

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

This mechanism relies solely on Boolean values that determine which hook macro to call. Those values are set within `\@nameauth@name` and `\AKA,` where they also affect branching. However, the name conditional macros act directly, based on the presence or absence of control sequences determined by the interplay of name arguments, the Boolean values checked above, and the control sequences generated by that interaction.

Perhaps a metaphor for this sort of "double interaction" could be "double-clutching" or, alternately, the difference between a manual and an automatic transmission. From the back-end side, you use the Boolean values and the name conditionals to perform a sort of "torque matching" between what goes in to the hook macros and what you want to happen when they complete. From the user side, this happens almost automatically, in a seemingly intuitive way with few additional keystrokes.

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

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

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

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

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

Here we parse names.

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

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

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

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

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

Handle first uses.

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

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

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

Handle first uses.

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

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

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

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

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

Handle first uses.

```
        \@nameauth@FirstFormattrue%
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNamefalse%
        \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}{}%
      \fi
    \else
```

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

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

Handle first uses.

```
          \@nameauth@FullNamefalse%
          \@nameauth@FirstNamefalse%
          \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}{}%
        \fi
      \fi
    \fi
  \else
```

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

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

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

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

Handle first uses.

```
        \@nameauth@FirstFormattrue%
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNamefalse%
        \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}{}%
      \fi
    \else
```

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

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

Handle first uses.

```
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNamefalse%
        \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}{}%
      \fi
    \fi
  \fi
```

Reset all the "per name" Boolean values.

```
\@nameauth@Lockfalse%
\@nameauth@InNamefalse%
\@nameauth@FirstFormatfalse%
\@nameauth@NBSPfalse%
\@nameauth@DoCapsfalse%
\@nameauth@Accentfalse%
\@nameauth@AllThisfalse%
\@nameauth@ShowCommafalse%
\@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}
```

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@DoFormatfalse}
```

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

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

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 and the index by setting up which kind we want to use. These will be inserted as appropriate as the output is formatted.

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

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

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

Here we parse names.

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

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

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

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

```
    \if@nameauth@RevThisComma
      \protected@edef\ISpace{,\space}%
      \protected@edef\Space{,\space}%
      \if@nameauth@NBSP
        \protected@edef\Space{,\nobreakspace}%
      \fi
    \fi
    \if@nameauth@RevThis
      \ex\@nameauth@FmtName\ex{\Reversed}%
    \else
      \ex\@nameauth@FmtName\ex{\SNNd}%
    \fi
    \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\ISpace\arge|see{\SNNb}}%
          \else
            \ifx\Suffb\@empty
              \@nameauth@Index{\csde}%
                {\ISNNd\ISpace\arge|see{\SNNb,\space\arga}}%
            \else
              \@nameauth@Index{\csde}%
              {\ISNNd\ISpace\arge|see{\Shortb,\space\arga,\space\Suffb}}%
            \fi
          \fi
        \fi
      \fi
    \fi
```

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

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

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

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

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

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

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

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

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

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

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

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

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

\UntagName This deletes an index 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@DoFormat` selects the type of name.

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

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

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

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

We make copies of the arguments to test them.

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

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

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

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

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\@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\ex\csgdef\ex\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

v0.7
General: Initial release 1
v0.75
\ForgetName: New argument added 87
\IndexName: Use current arguments 78
v0.8
General: Add features, bugfixes . . . 1
v0.85
\@nameauth@FmtName: Add comma suppression 57
\@nameauth@Name: Add comma suppression 58
General: Add package options 1
\AKA: Add comma suppression . . 69
\IndexName: Add comma suppression 78
\PName: Add comma suppression 75
\PName*: Add comma suppression 75
v0.86
General: Fix regressions 1
v0.9
\@nameauth@Suffix: Added 56
\@nameauth@TrimRoot: Suffix handling expandable 55
\@nameauth@TrimSuffix: Added 56
General: Add first name formatting; affix handling expandable 1
\AKA: Add starred mode; redesigned 69
\AKA*: Added 75
\FName: Added 69
\SubvertName: Added 88
v0.94
\@nameauth@FmtName: Add particle caps 57
\@nameauth@Index: Added 58
General: Add index suppression, error checking, name particle caps 1
\CapThis: Added 67
\ExcludeName: Added 82
\IndexActive: Added 68
\IndexInactive: Added 68
v0.95
\@nameauth@CRii: Added 56
\@nameauth@CapRoot: Added . . . 55
\@nameauth@FmtName: Works with microtype 57
General: Bugfixes 1
v0.96
\@nameauth@Name: Works w/ microtype, memoir 58
General: Bugfixes 1
v1.0
General: Works fully with microtype and memoir 1
v1.1
General: Bugfixes 1
v1.2
\TagName: Added 80
\UntagName: Added 81
v1.26
\@nameauth@CRii: Fixed 56
\AKA: Fix sorting of name suffixes 69
\IndexName: Fix name suffix sorting 78
v1.4
\@nameauth@Root: Made more robust 55
General: Add features, bugfixes . . . 1
\FName: Refactored 69
\FName*: Refactored 69
\Name*: Refactored 69
\ShowComma: Added 68
v1.5
\@nameauth@AllCapRoot: Added 56
\@nameauth@Name: Add reversing and caps 58
\@nameauth@TrimSuffix: Trim spaces 56
General: Add features, bugfixes, options 1
\AKA: Add reversing and caps . . . 69
\AllCapsActive: Added 67
\AllCapsInactive: Added 67
\CapName: Added 67
\RevComma: Added 67
\ReverseActive: Added 67
\ReverseCommaActive: Added . . 68
\ReverseCommaInactive: Added 68
\ReverseInactive: Added 67
\RevName: Added 67
v1.6
nameauth: Added 89
v1.7
General: Fix options processing . . 1
v1.8
General: Update docs 1
v1.9
\ForgetName: Ensure global undef 87
\KeepAffix: Added 68
\TagName: Fix cs collisions 80
\UntagName: Ensure global undef, fix cs collisions 81

v2.0
\@nameauth@FmtName: One macro instead of two 57
\@nameauth@Index: Redesigned tagging 58
\@nameauth@Name: Isolate malformed input; trim spaces; redesign tagging 58
\@nameauth@TrimRoot: trim spaces 55
General: Use dtxgen template instead of dtxtut; update docs . . 1
\AKA: Isolate malformed input; trim spaces; redesign tagging 69
nameauth: Redesigned argument handling 89
\ExcludeName: Isolate malformed input 82
\ForgetName: Isolate malformed input 87
\IndexActual: Added 68
\IndexName: Isolate malformed input; trim spaces; redesign tagging 78
\PretagName: Added 82
\SubvertName: Isolate malformed input 88
\TagName: Isolate malformed input; redesign tagging 80
\UntagName: Isolate malformed input; redesign tagging 81
v2.1
\@nameauth@CRiii: added 56
\@nameauth@CapRoot: Handle Unicode better 55
\@nameauth@Name: Isolate Unicode issues 58
General: Isolate Unicode issues . . . 1
\AccentCapThis: Added 67
\AKA: Isolate Unicode issues 69
v2.11
nameauth: Bugfix 89
v2.2
General: Add interface hooks and docs; fix bugs 1
\NameauthFName: Added 53
\NameauthName: Added 53
v2.3
\@nameauth@Name: Rename as internal macro 58
General: Improve docs and hooks; add features 1
\AKA: Expand starred mode 69
\ExcludeName: Distinguish excluded names from regular aliases 82
\ForgetName: Changed to allow global or local behavior 87
\GlobalNames: Added 68
\IfAKA: Added 86
\IfFrontName: Added 84
\IfMainName: Added 85
\LocalNames: Added 68
\Name: Change to interface macro 69
\NameauthLName: Added 53
\PName: Work directly with hooks 75
\SubvertName: Changed to allow global or local behavior 88
v2.4
\@nameauth@FmtName: Add hooks for non-formatted names 57
\@nameauth@Name: Define name CS after formatting; add token regs for hooks 58
General: Add text tagging features, add generic hooks, and prevent recursion 1
\AKA: Define name CS after formatting; add token regs for hooks 69
\FrontNameHook: Added 53
\GlobalNames: Ensured to be global 68
\IfAKA: Redesign exclusion test . 86
\LocalNames: Ensured to be global 68
\MainNameHook: Added 53
\NameAddInfo: Added 76
\NameClearInfo: Added 77
\NameQueryInfo: Added 76
v2.41
\@nameauth@Name: no local \newtoks 58
General: instantiate token registers only once 1
\AKA: no local \newtoks 69
nameauth: no local \newtoks . . . 89
v2.42
General: do not use \cmd in section headings 1

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.

Symbols

\@nameauth@Actual . 127
\@nameauth@AllCapRoot 96
\@nameauth@CRii 94
\@nameauth@CRiii . . . 95
\@nameauth@CapRoot . 78
\@nameauth@CheckDot 109
\@nameauth@Clean . . . 73
\@nameauth@EvalDot 111
\@nameauth@FmtName 113
\@nameauth@Index . . 128
\@nameauth@Name . . . 147
\@nameauth@Root 75
\@nameauth@Suffix . . 98
\@nameauth@TestDot 101
\@nameauth@TrimRoot 77
\@nameauth@TrimSuffix 100
\@nameauth@toksa . . *39*
\@nameauth@toksb . . *39*
\@nameauth@toksc . . *39*

A

\AccentCapThis . *17*, 468
Æthelred II, king . 20, 31
\AKA *26*, 495
\AKA* *26*, 761
\AllCapsActive . *16*, 471
\AllCapsInactive *16*, 470
Anaximander 19
Andreä, Johann 20
Antiochus IV Epiphanes, king 48
Arai Akino 16
Aristotle 4, 9, 11
Arouet, François-Marie . . . . . *see* Voltaire
Atatürk . . . . . *see* Kemal, Mustafa
Attila the Hun . . 4, 9, 11

B

Biermann, Johann* . . . 18

C

\CapName *16*, 469
\CapThis *17*, 467
Carnap, Rudolph 22, 23, 25
Carter, James Earl, Jr., president . . . 6, 32
Carter, Jimmy . . . . . *see* Carter, James Earl, Jr.
Chaplin, Charlie 38
Charles the Bald, emperor 13, 14
Chiang Kai-shek, president 15
Cicero, M.T. 13, 14
Clemens, Samuel L. . . . *see* Twain, Mark
Confucius 13, 14, 24
Cousot, Patrick 20

D

Dagobert I, king 12
de la Mare, Walter 17, 49
DE' MEDICI, Catherine 17
de Soto, Hernando 11, 17
Demetrius I Soter, king 47
Doctor Angelicus . . . *see* Thomas Aquinas
Dongen, Marc van . . 2, 58
Du Bois, W.E.B. 45
du Cange . . . . . *see* du Fresne, Charles
du Fresne, Charles . . . 34
DuBois, W.E.B. . . . . *see* Du Bois, W.E.B.

E

Einstein, Albert . . 13, 14
Elizabeth I, queen 4, 9, 11
environments:
  nameauth . . *10*, 1291
\ExcludeName . *33*, 1021

F

\FName *14*, 493
\FName* *14*, 494
\ForgetName . . *25*, 1184
Francis I, king 47
\FrontNameHook . . 27, *42*
FUKUYAMA Takeshi . 21

G

GARBO, Greta 20
\GlobalNames . . *25*, 486
Goethe, Johann Wolfgang von 49
Gossett, Louis, Jr. 15
Gregorio, Enrico 2
Gregory I, pope 28, 29, 31, 32
Gregory the Great . . . . *see* Gregory I

H

Hammerstein, Oskar, II 15, 19, 50
Harnack, Adolf 49
Hearn, Lafcadio 34
Henry VIII, king 12, 15, 48
Hope, Bob 8, 28
Hope, Leslie Townes . . . . *see* Hope, Bob
HOWELL, Thurston, III* 21

I

\if@nameauth@InAKA *39*
\if@nameauth@InName *39*
\IfAKA *24*, 1147
\IfFrontName . *23*, 1093
\IfMainName . . *23*, 1120
\IndexActive . . *30*, 488
\IndexActual . . *31*, 489
\IndexInactive . *30*, 487
\IndexName *30*, 848
Iron Mike *see* Tyson, Mike
Ishida Yoko 16

J

Jean sans Peur, duke . . 27
Jean the Fearless . . . *see* Jean sans Peur
John Eriugena 19

K

Kanno, Yoko† 16
\KeepAffix *15*, 482
Kemal, Mustafa . . 41, 44
Keynes, John Maynard 22
King, Martin Luther, Jr. 19
Koizumi Yakumo . . . *see* Hearn, Lafcadio
Konoe, Fumimaro, PM† 11

Kresge, Joseph *see* Kreskin, The Amazing
Kreskin, The Amazing 34

L

Lao-tzu 28, 29
Leo I, pope 31, 32
Leo the Great . . *see* Leo I
Lewis, Clive Staples 4, 8, 11, 14
Li Er . . . . . *see* Lao-tzu
`\LocalNames` . . . *25*, 485
Louis XIV, king . . 15, 28
Lueck, Uwe 2, 56
Luecking, Dan 35
Łukasiewicz, Jan 31

M

Maimonides . . . . . 28, *see also* Rambam
`\MainNameHook` . . . 26, *42*
Malebranche, Nicolas . 22
Mao Tse-tung, chairman 19, 48
Mill, J.S. 19
Moses ben-Maimon . . . . . *see* Maimonides

N

Nakano, Aiko† 16
`\Name` *13*, 491
`\Name*` *13*, 492
`\NameAddInfo` . . *26*, 767
`nameauth` (environment) . . . *10*, 1291
`\NameauthFName` . . *21*, 30
`\NameauthLName` . . *21*, 29
`\NameauthName` . . . *21*, 28
`\NameClearInfo` . *26*, 821
`\NameQueryInfo` . *26*, 794
`\NamesActive` . . *22*, 484
`\NamesFormat` . . . *21*, 25
`\NamesInactive` . *22*, 483
Nogawa Sakura 16

O

Oberdiek, Heiko . . . 2, 55

P

Patton, George S., Jr. . 46
Plato 47
`\PName` *29*, 762
`\PName*` 766
`\PretagName` . . . *31*, 982
Ptolemy I Soter, king . 48
Public, J.Q.* 46

R

Rambam . . . . . 28, *see also* Maimonides
`\RevComma` *19*, 475
`\ReverseActive` . *16*, 474
`\ReverseCommaActive` *19*, 479
`\ReverseCommaInactive` *19*, 477
`\ReverseInactive` *16*, 473
`\RevName` *16*, 472
Rockefeller, Jay . . . . . *see* Rockefeller, John David, IV
Rockefeller, John David, II 4, 8, 11
Rockefeller, John David, IV . . 8, 11, 14, 24
Rühmann, Heinrich Wilhelm . . . *see* Rühmann, Heinz
Rühmann, Heinz 27

S

Schlicht, Robert . . . 2, 20
Shikata Akiko 16
`\ShowComma` *15*, 481
Smith, John* . 32, 33, 46
Smith, John* (other) . . 32
Smith, John* (third) . . 32
Snel van Royen, Rudolph 28
Snel van Royen, Willebrord 28
Snellius . *see* Snel van Royen, Rudolph; Snel van Royen, Willebrord
Stephani, Philipp 2
Strietelmeier, John . . . 18
`\SubvertName` . *25*, 1238
Sullenberger, Chesley B., III 14, 30
Sun King . *see* Louis XIV
Sun Yat-sen, president 4, 15, 47

T

`\TagName` *31*, 913
Thomas à Kempis 17
Thomas Aquinas 29
Thomas of Aquino . . *see* Thomas Aquinas
Twain, Mark 29
Tyson, Mike 34

U

`\UntagName` *32*, 955

V

Vlad II Dracul 39
Vlad III Dracula 39
Vlad Ţepeş . . . . . *see* Vlad III Dracula
Vlad the Impaler . *see* Vlad III Dracula
Voltaire 29

W

Washington, George, president . . 4, 8, 11, 26, 41, 44
White, E. B. 18, 51
William I 29
William the Conqueror . . . . *see* William I

Y

Yamamoto Isoroku 4, 9, 11
Yohko 16
Yoshida Shigeru, PM . 12