# nameauth — Name authority management for consistency in text and index*

Charles P. Schaum†

Released 2016/11/01

**Abstract**

The nameauth package automates the correct formatting and indexing of names for professional writing. This aids the use of a **name authority** and the editing process without needing to retype name references.

## Contents



*This file describes version v3.03, last revised 2016/11/01.

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

# 1 Introduction

## 1.1 Preliminary Information

### Disclaimer

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

### Denotative Signs

In the index, fictional names have an asterisk (*). In this manual, "non-native" Eastern names are shown with a dagger (†). Names that use the older syntax are shown with a double dagger (‡). These marks are not added by the package macros and will not appear in users' works.

### Design

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

- **Automation** of name forms to aid professional writing. Move blocks of text and see the names reformat themselves.
  - Default to long name references first, then shorter ones.
  - Use alternate names only in the body text, not the index.
  - Perform name caps and reversing only in the body text.
- Name variants in the text still produce **consistent index entries**.
- One can design **complex name formatting**. Default is English typography, but other standards, such as "Continental" use of small caps, can be applied (Sections 2.4.7, 2.4.8, 2.7.5, and 2.11.7).
- One can **automate information retrieval** about names.
- One can implement and automate a **name authority**, a master list of names that permits known name variants.
- Some **cross-cultural naming conventions** are possible.
- **Automatic sort keys and tags** aid indexing.
- Provide **automatic name presentation** that could work in a LaTeX back-end for a document transform, database report, etc.

Indexing generally conforms to the standard in Nancy C. Mulvany, *Indexing Books* (Chicago: University of Chicago Press, 1994).

### Thanks

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

## 1.2 Technical Notes

About the package itself:

- Most current changes are compatible with older versions.
- The package works with `xindy` and `makeindex`. We recommend `xindy` for languages whose collating sequences do not map to English.[1]

**3.0** • Notable changes, features, and fixes that correspond to version numbers are indicated in the margin.

**3.0** • We support Eastern alternate names when using "native" format. We simplify indexing, custom formatting, and other tasks.

**3.0** • Name output, index page entries, and index cross-references are independent due to modular design.

**3.0** • Warnings for the indexing macros are suppressed unless one uses the `verbose` option. The `nameauth` environment will produce warnings.

**2.6** • The `comma` option and the old syntax are no longer restrictive, save with `\AKA` and its derivatives. See Sections 2.1.3, 2.4.1, and 2.10.

**2.5** • No formatting is selected by default. To implement formatting, see Sections 2.4.8, 2.6, and 2.11.5f.

About the manual:

- With the "dangerous bend" we show incompatibilities, complex topics, and points where caution is needed.
- This manual is designed to be compatible with both A4 and US letter stock size formats.
- Macro references are minimized for a "clean" index, showing how nameauth "normally" handles indexing.
- Some examples use package macros and internals in special ways to demonstrate a particular point. Usually we mention that with the example.

About package building:

- The nameauth package requires etoolbox, suffix, trimspaces, and xargs. The `dtx` file encoding is UTF-8; we cannot guarantee building and using this package on systems that are not Unicode-compliant.
- With each release, we test `nameauth` with dvi-mode `latex` and with pdf-mode engines `pdflatex`, `lualatex`, and `xelatex` using `makeindex`. We run the GNU Makefile with the ENGINE=⟨*engine*⟩ option.[2]
- This build used `pdflatex`. This item changes per LaTeX engine.
- This package is tested on Ubuntu Linux and Windows 7 (both vanilla TeX Live). Cygwin provides `make` on Windows. The `pdflatex` version of this package is released from the Ubuntu platform to CTAN.

[1] `\PretagName` may not be useful in that case. German *does* map to English: ä, ö, ü, and ß are ae, oe, ue, and ss. Norwegian *does not* map to English: æ, ø, and å come after z.

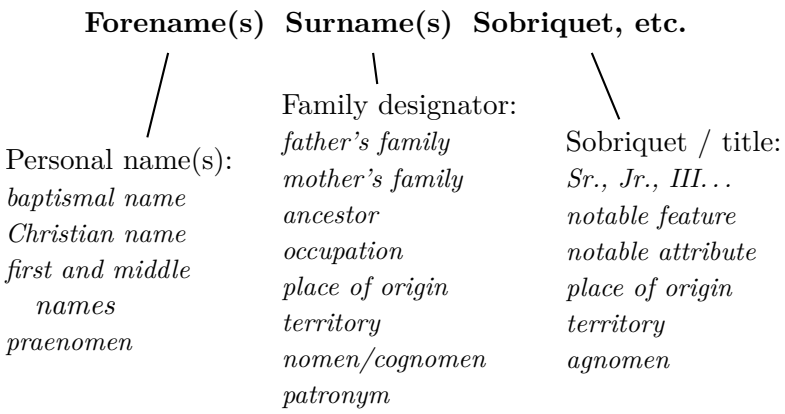
[2] The manual is used as the test suite. In dvi mode the manual omits all references to *TikZ* because some dvi display programs (*e.g.* `dviout`, but not `xdvi`) will emit errors about bad specials even if one just includes the tikz package. The *TikZ* diagrams herein will appear as blank space in that case. This does not affect nameauth proper.

## 1.3 What's In A Name?

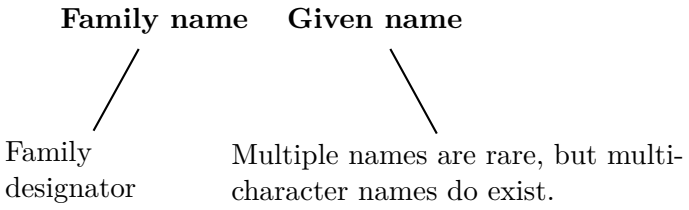
Name forms are ambiguous apart from historical and cultural contexts. In this manual we refer to three general classes of names, shown below. Section 2.1 shows how these classes are implemented.

Professional writing often calls for the full form of a person's name to be used when it first occurs, with shorter forms used thereafter. That is why, in each class, there is a required name and optional name elements.[3] Other naming systems can be adapted to these categories, *e.g.*, Icelandic, Hungarian, etc.

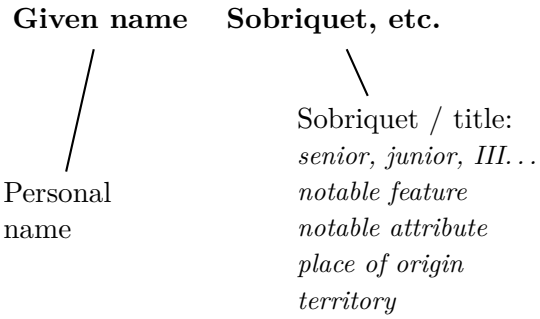
1. Western name:




2. Eastern name:




3. Ancient name:




Since we use these classes of names throughout the manual, it is helpful to become familiarized with this terminology and how one can apply it to one's own culture and context.

[3] Compare Mulvany, *Indexing Books*, pages 152–82, and the *Chicago Manual of Style*. That approach is adapted to LaTeX and its way of handling optional arguments.

# 2 Usage

## 2.1 Quick Start Guide

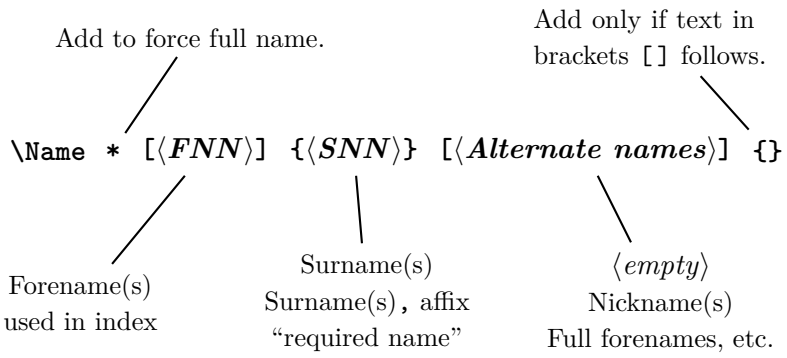
Here we offer templates for the most common macros and features. The details will come later. This section also formats the first uses of names in **boldface** to illustrate how first uses are long, but long uses need not be first.

### 2.1.1 Main Interface

We see how nameauth implements the classes of names, using `\Name` as an example (see also Section 2.3). The macros of this package halt with an error when:

- The ⟨*SNN*⟩ argument expands to the empty string or an ⟨*SNN, Affix*⟩ pair expands to ⟨*empty*⟩, ⟨*Affix*⟩.
- No shorthand is present for a name in the simplified interface (Section 2.1.2).

**Western Names**




**Examples:**

One always must include all fields for consistent index entries.

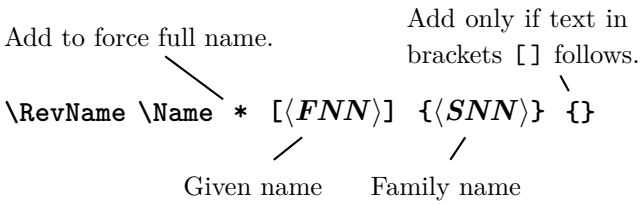
`\Name[George]{Washington}` . . . . . **George Washington**
`\Name[George]{Washington}` . . . . . Washington
`\Name[John David]{Rockefeller, II}` . . . . . **John David Rockefeller II**
`\Name[John David]{Rockefeller, II}` . . . . . Rockefeller

When using ⟨*Alternate names*⟩, the ⟨*FNN*⟩ argument must have a name in it. The ⟨*Alternate names*⟩ are swapped with the ⟨*FNN*⟩, but only in the body text:

`\Name [Clive Staples]{Lewis}` . . . . . **Clive Staples Lewis**
`\Name*[Clive Staples]{Lewis}[C.S.]` . . . . . C.S. Lewis
`\Name [Clive Staples]{Lewis}[C.S.]` . . . . . Lewis
`\Name*[Clive Staples]{Lewis}[Jack]` . . . . . Jack Lewis
`\Name [John David]{Rockefeller, IV}` . . . . . **John David Rockefeller IV**
`\Name*[John David]{Rockefeller, IV}[Jay]` . . . . . Jay Rockefeller IV
`\DropAffix\Name*[John David]{Rockefeller, IV}[Jay]` . . . Jay Rockefeller
`\Name [John David]{Rockefeller, IV}[Jay]` . . . . . Rockefeller

Western names cannot use the older syntax for affixes; see Sections 2.1.3 and 2.4.1. Notice how using nicknames or alternate names does not trigger an explicit first use. That is intentional. Compare Section 2.9.2 to "bend the rules."

### Eastern Names in the Text, Western Index Entry

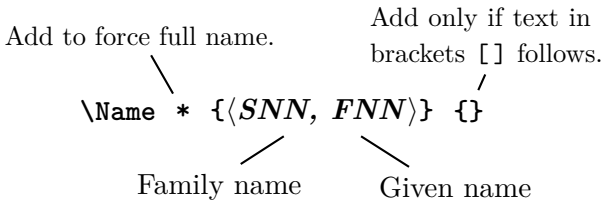



These are Western name forms without affixes that the reversing macros (Section 2.4.2) cause to display in Eastern order in the body text only. We indicate them with a dagger.

`\Name[Fumimaro]{Konoe}` . . . . . **Fumimaro Konoe**†
`\Name[Fumimaro]{Konoe}` . . . . . Konoe†
`\RevName\Name*[Fumimaro]{Konoe}` . . . . . Konoe Fumimaro†

The index entries are Western in fashion: ⟨*SNN*⟩, ⟨*FNN*⟩. This "non-native" form of Eastern names conflicts with both comma-delimited forms and the old syntax.

### Eastern Names in the Text, Eastern Index Entry



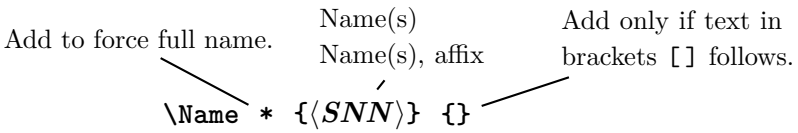

**Examples:**

The main feature of Eastern, ancient, and medieval name forms is the comma-delimited suffix. These forms have the same syntax, but different meanings.

`\Name{Yamamoto, Isoroku}` . . . . . **Yamamoto Isoroku**
`\Name{Yamamoto, Isoroku}` . . . . . Yamamoto
`\RevName\Name*{Yamamoto, Isoroku}` . . . . . Isoroku Yamamoto

These names truly are Eastern names. They always take the form ⟨*SNN FNN*⟩ in the index. See Section 2.4.2. We call this the "native" Eastern form. For the old syntax, see Section 2.1.3.

### Ancient Names




**Examples:**

These forms are for royalty and ancient figures. They have one or more personal names that may or may not have suffixes. For the older syntax see Section 2.1.3.

`\Name{Aristotle}` . . . . . **Aristotle**
`\Name{Aristotle}` . . . . . Aristotle
`\Name{Elizabeth, I}` . . . . . **Elizabeth I**
`\Name{Elizabeth, I}` . . . . . Elizabeth

## Macro Patterns:

Here we see a large subset of the nameauth macros in order to get a feel for their patterns. This may help one get used to them.

| | | |
|---|---|---|
| ⟨*prefix macros*⟩ | \Name | ⟨*arguments*⟩ |
| ⟨*prefix macros*⟩ | \Name* | ⟨*arguments*⟩ |
| ⟨*prefix macros*⟩ | \FName | ⟨*arguments*⟩ |
| | \IndexName | ⟨*arguments*⟩ |
| | \IndexRef | ⟨*arguments*⟩ ⟨*target*⟩ |
| | \ExcludeName | ⟨*arguments*⟩ |
| | \IncludeName | ⟨*arguments*⟩ |
| | \IncludeName* | ⟨*arguments*⟩ |
| | \PretagName | ⟨*arguments*⟩ ⟨*sort key*⟩ |
| | \TagName | ⟨*arguments*⟩ ⟨*tag*⟩ |
| | \UntagName | ⟨*arguments*⟩ |
| | \NameAddInfo | ⟨*arguments*⟩ ⟨*tag*⟩ |
| | \NameQueryInfo | ⟨*arguments*⟩ |
| | \NameClearInfo | ⟨*arguments*⟩ |
| | \IfMainName | ⟨*arguments*⟩ {⟨*y*⟩}{⟨*n*⟩} |
| | \IfFrontName | ⟨*arguments*⟩ {⟨*y*⟩}{⟨*n*⟩} |
| | \IfAKA | ⟨*arguments*⟩ {⟨*y*⟩}{⟨*n*⟩} |
| | \ForgetName | ⟨*arguments*⟩ |
| | \SubvertName | ⟨*arguments*⟩ |

## Prefix Macros:

| | |
|---|---|
| \CapThis | Capitalize first letter of ⟨*SNN*⟩ in body text.[4] |
| \CapName | Cap entire ⟨*SNN*⟩ in body text. Overrides \CapThis. |
| \RevName | Reverse name order in body text (for Eastern names). |
| \RevComma | Reverse Western names to ⟨*SNN*⟩, ⟨*FNN*⟩. Has no effect on native Eastern names. |
| \ShowComma | Add comma between ⟨*SNN*⟩ and ⟨*Affix*⟩. |
| \NoComma | Suppress comma between ⟨*SNN*⟩ and ⟨*Affix*⟩. Overrides \ShowComma. |
| \ForceFN | Force Eastern forename/ancient affix. Only affects non-Western forms. |
| \DropAffix | Drop name affix of Western name (in long name reference). Only affects Western names. |
| \KeepAffix | Insert non-breaking space between ⟨*SNN*⟩ and ⟨*Affix*⟩. Can prevent a native Eastern name from breaking. |
| \SeeAlso | Toggle cross-reference type from *see* to *see also* (with \IndexRef, \AKA, and \PName). |

Most ⟨*prefix macros*⟩ are "stackable" in front of the naming macros; for example, \CapName\RevName\Name....

[4] \AccentCapThis is a fall-back for when the nameauth package is used where system architecture or file encoding might cause errors with the automatic Unicode detection under NFSS.

### 2.1.2 Simplefied Interface

`nameauth` The `nameauth` environment can replace `\Name`, `\Name*`, and `\FName` with name shorthands that inter-operate with the other package macros.

Although not required, `nameauth` is best used in the document preamble to avoid undefined control sequences.[5] The italicized comments below are not part of the example proper. Macro fields have uniform widths only to help compare argument types.

```
\begin{nameauth}
  \< ⟨cseq1⟩ & ⟨FNN⟩ & ⟨SNN⟩          & >        % West./Non-native East.
  \< ⟨cseq2⟩ & ⟨FNN⟩ & ⟨SNN, affix⟩   & >                 % Western
  \< ⟨cseq3⟩ &       & ⟨SNN⟩          & >              % ancient/mono
  \< ⟨cseq4⟩ &       & ⟨SNN, affix⟩   & >              % royal/ancient
  \< ⟨cseq5⟩ &       & ⟨SNN, FNN⟩     & >            % Native Eastern
  \< ⟨cseq6⟩ &       & ⟨SNN⟩          & ⟨FNN or affix⟩ >   % old syntax[6]
\end{nameauth}
```

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

- In this context, "`\<`" is an escape character and a control sequence. If you forget it or just use `<` without the backslash, you will get errors.
- There *must* be four argument fields (three ampersands) per line. Leaving out an ampersand will cause an error. Think "holy hand grenade of Antioch" from *Monty Python and the Holy Grail*.
- Extra spaces in each `&`-delimited field are stripped, as is also the case in the main interface (Section 2.5).
- Include trailing braces `{}` or the like if you must prevent subsequent text in brackets `[]` from being seen as an optional argument.
- The old syntax (Section 2.1.3), triggered by an empty ⟨*FNN*⟩ field, causes the ⟨*Alt. names*⟩ field to be interpreted as either ⟨*Eastern FNN*⟩ or ⟨*affix*⟩.

**3.0** Current versions permit alternate names for Eastern names and Western names, but not mononyms. **To switch between ⟨*FNN*⟩ and ⟨*Alt. names*⟩, simply use the forms above with an optional argument after the shorthand.** The forms below only print ⟨*Alt. names*⟩, never ⟨*FNN*⟩, in the body text. See also Section 2.3.2 and the example on the next page.

```
\begin{nameauth}
  \< ⟨cseq7⟩  & ⟨FNN⟩ & ⟨SNN⟩          & ⟨Alt. names⟩ >    % Western alt.[7]
  \< ⟨cseq8⟩  & ⟨FNN⟩ & ⟨SNN, affix⟩   & ⟨Alt. names⟩ >     % Western alt.
  \< ⟨cseq9⟩  &       & ⟨SNN, affix⟩   & ⟨Alt. names⟩ >     % Ancient alt.
  \< ⟨cseq10⟩ &       & ⟨SNN, FNN⟩     & ⟨Alt. names⟩ >     % Eastern alt.
\end{nameauth}
```

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

[6] See Section 2.1.3.

[7] Instead of this form you could use `\IndexRef` or `\AKA` to create a cross-reference, for example, from Jay Rockefeller to John David Rockefeller IV. See Sections 2.7.4 and 2.10.

This example of the `nameauth` environment uses a fairly complete set of names and illustrates how extra spaces are ignored. We also see the shorthands used in the body text (and the index).

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

Below, "non-native" Eastern name forms are shown with a dagger (†). Please see Section 2.4.2 to avoid pitfalls with Eastern names and reversing macros.

Western:
`\Wash` . . . . . **George Washington**
`\LWash` . . . . . George Washington
`\Wash` . . . . . Washington
`\SWash` . . . . . George
`\RevComma\LWash`
. . . . . Washington, George

Particles: (Section 2.4.6)
`\Soto` . . . . . **Hernando de Soto**
`\Soto` . . . . . de Soto
`\CapThis\Soto` . . . . . De Soto

Affixes: (Section 2.4.1)
`\JRII` . **John David Rockefeller II**
`\LJRII` . . . . John David Rockefeller II
`\DropAffix\LJRII`
. . . . . John David Rockefeller
`\JRII` . . . . . Rockefeller
`\SJRII` . . . . . John David

Nicknames: (Section 2.3.2)
`\JRIV` **John David Rockefeller IV**
`\LJRIV[Jay]` . . . . Jay Rockefeller IV
`\SJRIV[Jay]` . . . . . Jay
`\DropAffix\LJayR` . . Jay Rockefeller
`\SJayR` . . . . . Jay
`\LewisFull` . . **Clive Staples Lewis**
`\LLewis` . . . . . C.S. Lewis
`\Lewis` . . . . . Lewis
`\SLewis` . . . . . C.S.

Ancient / Mononym
`\Aris` . . . . . **Aristotle**
`\Aris` . . . . . Aristotle

Medieval/Royal:
`\Eliz` . . . . . **Elizabeth I**
`\Eliz` . . . . . Elizabeth
`\Attil` . . . **Attila the Hun**
`\Attil` . . . . . Attila

"Non-native" Eastern:
`\Konoe` **Fumimaro Konoe**†
`\LKonoe` . Fumimaro Konoe†
`\Konoe` . . . . . Konoe†
`\SKonoe` . . . . . Fumimaro†
`\CapName\RevName\LKonoe`
. . . . . KONOE Fumimaro†
`\CapName\Konoe` . KONOE†

"Native" Eastern:
`\CapName\Yamt`
. **YAMAMOTO Isoroku**
`\CapName\LYamt`
. . . . YAMAMOTO Isoroku
`\CapName\Yamt`
. . . . . YAMAMOTO
`\RevName\LYamt`
. . . . . Isoroku Yamamoto
`\SYamt` . . . . . Yamamoto
`\ForceFN\SYamt`
. . . . . Isoroku

Sections 2.4.6, 2.4.7, and 2.7.5 deal with the pitfalls of accents and capitalization, as well as why you should use `\PretagName` for any name with control sequences or active Unicode under NFSS. This becomes very important when authors and publishers use medieval names as Western names.

When index tagging or pre-tagging names, the ⟨*Alternate names*⟩ field has no effect on index tags because it appears only in the text. `\JRIV` and `\JayR` need only one tag, as do `\LewisFull` and `\Lewis`:

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

### 2.1.3 Older Syntax

**3.0** An older syntax remains for backward compatibility with early versions of nameauth. The old syntax generally inter-operates with the new, but it limits the use of `\AKA` and its derivatives. It does not permit using any name forms with comma-delimited suffixes. See also Section 2.13.

```
\Name{Henry}[VIII]                          % royal name
\Name{Chiang}[Kai-shek]                     % Eastern name
\begin{nameauth}
  \< Dagb & & Dagobert & I >                % royal name
  \< Yosh & & Yoshida & Shigeru >           % Eastern name
\end{nameauth}
```

Since the ⟨*FNN*⟩ fields are empty, the final field becomes either ⟨*affix*⟩ or ⟨*FNN*⟩ and will appear in the index. With this example, we have:

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

**2.6** `\Name{Henry}[VIII]` (old syntax) will share name occurrences, tags, and index entries with `\Name{Henry, VIII}` (new syntax), as we see below. To avoid confusion, we do not recommend mixing these forms.

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

**3.0** Presently `\Name*{Henry, VIII}[Tudor]` prints "Henry Tudor" in the body text and "Henry VIII" in the index. In former versions of this package it would have produced "Henry VIII Tudor" in the text and in the index. **Since the older behavior was discouraged, it no longer will be supported and no backward compatibility will exist.** See also Sections 2.7.6 and 2.8.

## 2.2 Package Options

One includes the nameauth package thus:

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

The options have no required order. Still, we discuss them from the general to the specific, as the headings below indicate.

**Default options are shown in boldface.**

### Choosing Features

#### Enable Package Warnings

| | |
|---|---|
| verbose | Show warnings about index cross-references. |

**3.0** The default suppresses package warnings from the indexing macros. Warnings from the nameauth environment are not suppressed.

#### Choose Formatting

| | |
|---|---|
| **mainmatter** | **Start with "main-matter names" and formatting hooks (page 13).** |
| frontmatter | Start with "front-matter names" and hooks. |
| alwaysformat | Use only respective "first use" formatting hooks. |
| formatAKA | Format names declared by \AKA like other main- and front-matter names. |
| oldAKA | Force \AKA* to act like it did before v.3.0. |

The default mainmatter option and the frontmatter option enable two different systems of name use and formatting. They are mutually exclusive. \NamesActive starts the main matter system when frontmatter is used. See Section 2.6.

The alwaysformat option forces "first use" hooks globally. The formatAKA option affects "first-use" formatting and \AKA (Section 2.10). Both alwaysformat and formatAKA can be used with either mainmatter or frontmatter.

**3.0** **Using oldAKA option forces \AKA* always to print a "forename" field in the text, as it did in versions 2.6 and older.** Otherwise the current behavior of \AKA* prints in the same fashion as \FName (see Sections 2.3.2 and 2.10).

#### Enable/Disable Indexing

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

These apply only to the nameauth package macros. The default index option enables name indexing right away. The noindex option disables the indexing of names until \IndexActive enables it. **Both noindex and \IndexInactive suppress index tags.** See Section 2.7.1.

#### Enable/Disable Index Sorting

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

The default allows \PretagName to create sort keys used with NFSS or makeindex and its analogues. This option exists to switch sorting off if needed for xindy. See Section 2.7.5.

## Affect the Syntax of Names

### Show/Hide Affix Commas

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

This option is set at load time. If you use *modern standards*, choose the default `nocomma` option to get, *e.g.*, "James Earl Carter Jr." If you need to adopt *older standards* that use commas between surnames and affixes, you have two choices:

1. The `comma` option globally produces, *e.g.*, "James Earl Carter, Jr."
2. Section 2.4.1 shows how one can use `\ShowComma` with the `nocomma` option and `\NoComma` with the `comma` option to get per-name results.

### Capitalize Entire Surnames

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

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

Still, you can use this package with non-English conventions. You can add, *e.g.*, uppercase or small caps in surnames, formatting them also in the index. See also Sections 2.4.8 and 2.11.7 The simplified interface aids the embedding of control sequences in names. Section 2.4.2 deals with capitalization on a section-level and per-name basis.

### Reverse Name Order

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

These options should not be used indiscriminately. Reversing can have unwanted results with ancient/medieval names. Section 2.4.2 can help guide you on whether to consider global or per-name reversing for Eastern names.

So-called "last-comma-first" lists of names via `allrevcomma` and the macros `\ReverseCommaActive` and `\RevComma` (Section 2.4.5) are *not* the same as the `comma` option. They are designed for Western names.

## Typographic Post-Processing

Sections 2.4.8, 2.6, and 2.11.5f. explain this topic in greater detail, but are more advanced regarding technical details. Post-processing does not affect the index and comes after "syntactic formatting" (control sequences always present in the nameauth macro arguments).[8]

**2.4** What was "typographic formatting" has become a generalized concept of "post-processing" via these hook macros:

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

Sections 2.6 and 2.11.5f. offer substantially more complex possibilities for such hooks.[9] By default, they do nothing. Changes to the formatting hooks apply within the scope where they are made. To change that, use `\global` explicitly.

`\global`

English typography has been the default design choice for this package. Still, one can use German, French, and similar standards, which I group as "Continental." Sections 2.4.8 and 2.11.7 have more on the topic, as well as the use of sort tags in Section 2.7.5. Continental standards format surnames only, both in the text and in the index. This conflicts with some deliberately ambiguous name forms in nameauth.[10] To get formatting in the index one must add it in the macro arguments. The simplified interface aids this process. Continental users may have to implement their own capitalization features; see Sections 2.4.6 and 2.11.7.

### Formatting Attributes

| | |
|---|---|
| `smallcaps` | Set the first use of an entire name in small caps.[11] |
| `italic` | Set the first use of an entire name in italic. |
| `boldface` | Set the first use of an entire name in boldface. |
| `noformat` | **Do not define a default format.** |

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

**2.5 Current versions assign no default formatting to names.**

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

[9] 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.

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

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

## 2.3 Naming Macros

### 2.3.1 \Name and \Name*

\Name
\Name*

\Name displays and indexes names. Also see the quick start (Section 2.1.2) and reference (Section 2.12). It always prints the required "surname" field. \Name prints the full name at the first occurrence, then a partial form thereafter. \Name* always prints the full name. These macros generate index entries before and after a name in the body text in case of a page break. The general syntax is:

\Name [⟨*FNN*⟩]{⟨*SNN, opt. FNN/Affix*⟩}[⟨*Alternate names*⟩]
\Name*[⟨*FNN*⟩]{⟨*SNN, opt. FNN/Affix*⟩}[⟨*Alternate names*⟩]

**3.0** In the body text, not the index, the ⟨*Alternate names*⟩ field replaces the ⟨*FNN*⟩ field or the ⟨*opt. FNN/Affix*⟩ field if they exist. If neither of the latter exist, then the old syntax is used (Section 2.1.3). We see this below:

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

| | |
|---|---|
| \Name [Albert]{Einstein} or \Einstein | Albert Einstein |
| \Name*[Albert]{Einstein} or \LEinstein | Albert Einstein |
| \Name [Albert]{Einstein} or \Einstein | Einstein |
| \Name [M.T.]{Cicero} or \Cicero | M.T. Cicero |
| \Name*[M.T.]{Cicero}[Marcus Tullius] | Marcus Tullius Cicero |
| \Name [M.T.]{Cicero} or \Cicero | Cicero |
| \Name {Confucius} or \Confucius | Confucius |
| \Name {Miyazaki, Hayao} or \Miyazaki | Miyazaki Hayao |
| \Name*{Miyazaki, Hayao}[Sensei] | Miyazaki Sensei |
| \Name {Miyazaki, Hayao} or \Miyazaki | Miyazaki |
| \Name {Charles, the Bald} or \CBald | Charles the Bald |
| \Name*{Charles, the Bald} or \LCBald | Charles the Bald |
| \Name {Charles, the Bald} or \CBald | Charles |

When using the simplified interface, the preferred way to get alternate names is \LCicero[Marcus Tullius] and \LMiyazaki[Sensei]: Marcus Tullius Cicero and Miyazaki Sensei. The next section explains why that is so.

Note also that the alternate forename goes away in subsequent name references. \Name [M.T.]{Cicero}[Marcus Tullius] shows up as just Cicero in that case. By default, subsequent name references are surnames only.

### 2.3.2 Forenames: \FName

\FName
\FName*

\FName and its synonym \FName* print just personal names, but only in subsequent name uses. The two macros are the same in case you edit \Name* by adding an F to get a first reference. They print a full name, not a short name, when a name is used for the first time. The syntax is:

\FName[⟨*FNN*⟩]{⟨*SNN, opt. FNN/Affix*⟩}[⟨*Alternate names*⟩]

**3.0**
\ForceFN

These macros work with both Eastern and Western names, but to get an Eastern personal name, one must precede these macros with \ForceFN. Otherwise you would get poor results with some royal and ancient names. See also Sections 2.9.2 and 2.12. Examples of general use include:

| | |
|---|---|
| \FName[Albert]{Einstein} or \SEinstein | Albert |
| \FName[M.T.]{Cicero}[Marcus Tullius] or \SCicero[Marcus Tullius] | Marcus Tullius |
| \FName{Confucius} or \SConfucius | Confucius |
| \FName{Miyazaki, Hayao} or \SMiyazaki | Miyazaki |
| \ForceFN\FName{Miyazaki, Hayao} or \ForceFN\SMiyazaki | Hayao |
| \ForceFN\FName{Miyazaki, Hayao}[Sensei] or \ForceFN\SMiyazaki[Sensei] | Sensei |
| \FName{Charles, the Bald} or \SCBald | Charles |
| \ForceFN\FName{Charles, the Bald} or \ForceFN\SCBald | the Bald |

If we use \FName[Chesley B.]{Sullenberger, III}[Sully] we get "Sully Sullenberger III" and "Sully" because the ⟨*Alternate names*⟩ always replace the forenames. Please use caution!

```
\begin{nameauth}
  \< LewisFull & Clive Staples & Lewis & >
  \< Lewis & Clive Staples & Lewis & C.S. >
  \< Ches & Chesley B. & Sullenberger, III & >
  \< Sully & Chesley B. & Sullenberger, III & Sully >
\end{nameauth}
```

Some names might call for this name-swapping. For example, if a book section refers always to C.S. Lewis, but another section introduces him as Clive Staples Lewis and the editor wants to index him as such, then the two shorthands for Lewis above might be appropriate. Section 2.1.2 illustrates several examples.

Using "default nicknames" in the simplified interface has some caveats. 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.]." With \Sully, the ⟨*Alternate names*⟩ field is already occupied, making [Chesley B.] normal text.

## 2.4 Language Issues

### 2.4.1 Affixes Need Commas

Comma-delimited affixes handle several different name types. *Always include a comma as an affix delimiter*, even when commas are not printed. Extra spaces between the comma and affix are ignored.

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

One cannot use the old syntax with the Hammerstein example. If you tried to use `\Name[Oskar]{Hammerstein}[II]` you would get "II Hammerstein." Western names with suffixes must use the comma-delimited syntax. Also, one must use comma-delimited suffixes to cross-reference all name forms with `\AKA`. For a full description see Section 2.10.

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

`\DropAffix` **3.0** Preceding the naming macros with `\DropAffix` will suppress an affix in a Western name. `\DropAffix\Name*[Oskar]{Hammerstein, II}` produces "Oskar Hammerstein." This does not affect non-Western names.

One can switch between the macros below without affecting entries in the index. Do, however, strive for consistency.

`\ShowComma` If you do not want to use the `comma` option, `\ShowComma` gets the same results on a per-name basis for Western names while using the default `nocomma`:

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

`\NoComma` **2.6** If you do wish to use the `comma` option, `\NoComma` can suppress a comma between the surname and affix on a per-name basis. For example, we alter package internals to simulate the `comma` option for the two tables below:

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

This might be useful when making a comma-reversed form (Section 2.4.5) under the `comma` option, where you do not want the form ⟨*SNN*⟩, ⟨*affix*⟩, ⟨*FNN*⟩:

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

### 2.4.2 Eastern Names

non-native

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

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

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

native

In contrast, there are two general forms of syntax for "native" Eastern name forms, which are indexed as such and appear in Eastern name order in the body text. The new syntax permits alternate names; the old does not:

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

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

\ReverseActive
\ReverseInactive
\RevName

In addition to the class options (Section 2.2), the macros `\ReverseActive` and `\ReverseInactive` toggle reversing on a larger scale, while `\RevName` is used once per name. The reverse output mechanism is designed to reverse full names only. Nevertheless, it does not force full names. Results vary, based on the type of Eastern name forms being used. Non-native forms are shown by a dagger (†):

| | *unchanged* | `\RevName` |
|---:|---|---|
| `\LKonoe` | Fumimaro Konoe† | Konoe Fumimaro† |
| `\LKonoe[Minister]` | Minister Konoe† | ⟨*not appropriate*⟩ |
| `\Konoe` | Konoe† | Konoe† |
| `\SKonoe` | Fumimaro† | Fumimaro† |
| `\LYamt` | Yamamoto Isoroku | Isoroku Yamamoto |
| `\LYamt[Admiral]` | ⟨*not appropriate*⟩ | Admiral Yamamoto |
| `\Yamt` | Yamamoto | Yamamoto |
| `\SYamt` | Yamamoto | Yamamoto |
| `\ForceFN\SYamt` | Isoroku | Isoroku |

**3.0** Creating "last-comma-first" listings by surname (Section 2.4.5) only makes sense with Western names and maybe non-native Eastern names, but not with native Eastern names or ancient forms. That is why native Eastern forms and ancient forms are unaffected by the commatized form of reversing.

\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.

\AllCapsActive
\AllCapsInactive
\CapName

The nameauth package allows one to capitalize entire family names in the body text while keeping them in standard English form in the index. This capitalization is designed to work with Eastern name forms. Use `\AllCapsActive`, `\AllCapsInactive`, and `\CapName` for fully-capitalized family names in the body text. These macros are analogous to the reversing macros and may be used alone or with other prefix macros, *e.g.*: `\CapName\RevName\Name*{`⟨*SNN, Affix*⟩`}`.

\global

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

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

| | *unchanged* | `\CapName\RevName` |
|---|---|---|
| `\Name*[Yoko]{Kanno}` | Yoko KANNO† | KANNO Yoko† |
| `\Name*{Arai, Akino}` | ARAI Akino | Akino ARAI |
| `\Name*{Ishida}[Yoko]` | ISHIDA Yoko‡ | Yoko ISHIDA‡ |
| `\Name*{Yohko}` | YOHKO | YOHKO |

Capitalization and reversing precede post-processing. The reversing and capitalization macros also work with `\AKA`. They affect only the text, not the index. For caps in the text and index see Sections 2.4.8 and 2.11.7.

### 2.4.3 Initials

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

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

| | |
|---|---|
| `\White` and `\LWhite` | E. B. White |
| Normal text: | E. B. White |

### 2.4.4 Hyphenation

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

```
\begin{nameauth}
  \< Striet & John & Striet\-el\-meier & >
\end{nameauth}
```

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

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

### 2.4.5 Listing by Surname

\ReverseCommaActive
\ReverseCommaInactive
\RevComma

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

**3.0** These macros work independently of the reversing macros used with Eastern names. This change allows reversing and commas to work harmoniously with "native" Eastern and ancient names:

| | | |
|---|---|---|
| John Stuart Mill | Mill, John Stuart | OK |
| Oskar Hammerstein II | Hammerstein II, Oskar | OK |
| John Eriugena | John Eriugena | no change |
| Mao Tse-tung | Mao Tse-tung | no change |
| Confucius | Confucius | no change |

**3.0** Since reversing with commas is independent of "native" Eastern and ancient names, we see its effects on "non-native" Eastern names:

| | |
|---|---|
| `\LKonoe` | Fumimaro Konoe† |
| `\RevName\LKonoe` | Konoe Fumimaro† |
| `\RevComma\LKonoe` | Konoe, Fumimaro† |

\global

Both `\ReverseCommaActive` and `\ReverseCommaInactive` have the same local restrictions as the other state-changing macros unless you use `\global`.

### 2.4.6 Particles

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

non-breaking spaces

We recommend inserting `~` or `\nobreakspace` between particles and names if the particles are less than three letters. This will keep them from becoming lost

**3.0** because of a bad break. `\CapThis` (below) no longer consumes the space between a single-letter particle and a name.

Some particles look very similar. For example, *L'* and *d'* are two separate glyphs each, while *Ľ* and *ď* are one Unicode glyph each.

\CapThis

In English, these particles go in the ⟨*SNN*⟩ field of `\Name`, *e.g.*, Walter de la Mare. When the last name with the particle appears at the beginning of a sentence, one must capitalize the particle:

```
\CapThis\Name[Walter]{de la Mare} will think it fair.%
\CapThis\Soto\ would agree.
```

De la Mare will think it fair. De Soto would agree.

The Continental style for surnames (Sections 2.4.8 and 2.11.7) does not work with `\CapThis`. Consider the example below or modify the examples from Section 2.11.7 to implement an alternate method of capitalization.

| | |
|---|---|
| `\Name[Catherine]{\textsc{de'~Medici}}` | Catherine de' Medici |
| `\textsc{De'~Medici}%` | |
| `\IndexName[Catherine]{\textsc{de'~Medici}}` | De' Medici |

`\AccentCapThis` **3.0**

If the source files for the nameauth package have Unicode encoding and run on a Unicode-compliant system, `\AccentCapThis` is not necessary. See also page 63. If the text encoding of the source files is changed or there are system encoding issues, `\AccentCapThis` might be needed with NFSS when the first name character is an active Unicode character. See also Section 2.11.3.

Medieval name issues

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

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

The medieval forms Thomas à Kempis and Thomas use the particle as the first part of an affix. Please do not confuse the medieval forms with the Western forms. Otherwise you will get similar names with different index entries.[12]

Many people still use medieval affixes as Western surnames: `\CapThis\KempW` displays "À Kempis."[13] The publisher's way of handling names may differ from the standard way. This package allows for such variations. Yet some publishers have problems with some name forms.[14] Developing a good rapport with the publisher will help you apply this package to the company's style.

Alternates

If the nameauth package is not used with UTF-8 encoding under NFSS or with `xelatex`/`lualatex`, the package still will work. Instead use control sequences, like `\Name[Thomas]{\'a~Kempis}`.

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 2.12.

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

[13] Ethnocentric treatment of names still occurs in academic literature. This is due to simplicity in work flow or conformity to name authorities. The nameauth package accommodates that, even if the package author finds such practices to be culturally insensitive.

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

### 2.4.7 Accented Names

For names that contain accented characters, using `xelatex` or `lualatex` with `xindy` (`texindy`) is recommended. See also Section 2.11.4.

In NFSS, accented characters are active. Especially when using `makeindex`, use `\PretagName` for all names with active Unicode characters (Sections 2.7.5 and 2.11.3). These active characters differ from the control sequences you type:

| | |
|---|---|
| `\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 2.11.3 on how to add additional Unicode glyphs to the default set under NFSS, inputenc, and fontenc. One may use expandable control sequences in names (thanks Robert Schlicht). Also, you can use `\edef` and `\noexpand` in names (Section 2.11.7, thanks Patrick Cousot).

### 2.4.8 Non-English Format

Continental small caps

See Sections 2.6 and 2.11.7 in addition to this section. We place this section here because it has a language-based element.

By default, name post-processing only affects the text. Also by default, you cannot post-process Continental formatting because it occurs in both the text and the index via control sequences in the nameauth macro arguments. That usually requires `\PretagName` for index sorting (Section 2.7.5). Section 2.11.7 explains how to implement post-processing via `\noexpand` and other steps.

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

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

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

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

A comma delimiter will split the macro argument into a root and an affix. To avoid an error regarding unbalanced braces, format the name and suffix separately. A suffixed Western name with Continental formatting looks like:

```
\PretagName[Thurston]{\textsc{Howell},\textsc{III}}%
  {Howell, Thurston 3}
\begin{nameauth}
  \< Howell & Thurston & \textsc{Howell},\textsc{III} & >
\end{nameauth}
```

`\Howell` generates Thurston Howell III, then Howell. The old syntax cannot be used with Thurston Howell III.

For Eastern names we have a related example that uses both the new syntax and the old syntax:

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

| | |
|---|---|
| `\Fukuyama` | FUKUYAMA Takeshi |
| `\OFukuyama` | FUKUYAMA |
| `\LOFukuyama` | FUKUYAMA Takeshi |
| `\Fukuyama` | FUKUYAMA |

The old syntax and new syntax inter-operate. The built-in capitalization macros will not work here. For an alternate solution, see Section 2.11.7.

## 2.5 Spaces and Punctuation

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

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

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

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

Grouping tokens can frustrate this: `{\Name*[Martin Luther]{King, Jr.}}.` produces "Martin Luther King Jr.." (two periods). Enclosing `{Jr.}` within braces or a macro will do the same. If you must format the affix, leave the period outside: `\Name[Martin Luther]{\textsc{King}, \textsc{Jr}.}` Cf. Section 2.4.8.

## 2.6 Formatting in the Text

There are two kinds of formatting at work:

1. **Syntactic Formatting:** The default includes reversing and capitalizing macros. Otherwise one must embed control sequences in name arguments.
2. **Name Post-Processing:** The hook macros apply formatting to the final form that a name takes in the text.

\NamesFormat
\FrontNamesFormat
\MainNameHook
\FrontNameHook
**2.5**

The "main-matter" and "front-matter" systems are independent systems of formatting and first/subsequent name use. The main-matter system uses `\NamesFormat` to post-process first occurrences of names and `\MainNameHook` for subsequent uses. The front-matter system uses `\FrontNamesFormat` for first occurrences of names and `\FrontNameHook` for subsequent uses. The `alwaysformat` option causes every name to be formatted with the "first-use" hooks.

\NamesActive
\NamesInactive

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

\global

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

By default, these two systems of names differ only between recognition of first and subsequent uses. We change the first-use formatting hooks in this section to show a greater distinction:

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

Now we switch to the "front matter" mode:

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

Then we switch back to "main matter" mode:

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

Below we simulate the `alwaysformat` option by manipulating the package internals. After the examples, we reset the formatting hooks.

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

Basic formatting changes can take either the font switch forms or the font command forms. The following also work:

```
\renewcommand*\NamesFormat{\textsc}
\renewcommand*\FrontNamesFormat{\textbf}
```

That is because the formatting hooks are called in such a manner that lets them either have one argument or not and keeps their changes local via:

```
\bgroup⟨Hook⟩{#1}\egroup
```

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

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

The problem above is that John Maynard Keynes in the text affects formatting in the footnotes.[15] Using `\ForgetName` to address this can be tedious.

A different, simpler solution uses the front-matter system:

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

Your footnotes do not affect the main body text now.[16] We change footnotes back to normal and restore the formatting hooks with the following:

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

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

[15] You get the name `\Name[John Maynard]{Keynes}` Keynes instead of `\ForgetName[John Maynard]{Keynes}\Name[John Maynard]{Keynes}` *John Maynard Keynes*.

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

## 2.7 Indexing Macros

**3.0** Current versions of nameauth offer greater flexibility with indexing but still implement some error protection. We cover the indexing macros here because the later macros in this manual build on many of their concepts.

### 2.7.1 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. They work differently than `\ExcludeName` and `\IncludeName`, which leverage the cross-referencing system. **`\IndexInactive` also suppresses index sorting and tagging macros.**

\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.

### 2.7.2 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.

### 2.7.3 Index Entries

\IndexName

The naming macros (`\Name`, etc.) use this macro to create index entries. Direct use of `\IndexName` is available also to users. 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*⟩ for Western and native Eastern name forms. Otherwise, if ⟨*FNN*⟩ are absent, `\IndexName` sees ⟨*Alternate names*⟩ as an affix or Eastern forename using the old syntax.

If used after `\IndexInactive` this macro does nothing until `\IndexActive` appears. It will not create index entries for names used as cross-references by `\IndexRef` and `\AKA`. This provides a basic level of error protection.

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.

### 2.7.4 Index Cross-References

\IndexRef
**3.0**

The cross-referencing macros (\AKA, etc.) use this macro. Also available to users, \IndexRef creates a *see* reference by default from the name defined by its first three arguments to whatever one puts in the final argument. The syntax is:

\IndexRef[⟨*FNN*⟩]{⟨*SNN*⟩}[⟨*Alternate names*⟩]{⟨*reference target*⟩}

The name used for the cross-reference is parsed in the same way as by \IndexName. The final argument is neither parsed nor checked to see if a corresponding main entry exists. For example, to cross-reference "Sun King" with Louis XIV use: \IndexRef{Sun King}{Louis XIV}.

**You are encouraged to see page 38 for handling cases of complex cross-references with either \AKA or \IndexRef.**

\SeeAlso
**3.0**

One can precede \IndexRef, \AKA, or \PName with \SeeAlso to produce a *see also* reference for a name that has appeared already in the index.[17] However, this should be used with caution, as the following points indicate:

- Once a cross-reference is created, that name cannot be used for page number entries. "Bar. . . 10, *see also* Foo" means that one cannot have an entry for "Bar" after page 10.
- The target of a reference, however, remains available. "Bar. . . 10, *see also* Foo" means that an entry for "Foo. . . 12, 13" is okay.
- A *see* reference has no page references before it. "Bar. . . 10, *see* Foo" is an invalid index entry.
- A *see also* reference appears after extant page references. "Bar. . . 10, *see also* Foo" is just fine, but "Bar. . . 10, *see also* Foo, 11, 14" is invalid.
- Having *see* and *see also* references on different index entry lines is suppressed. One can use \IndexRef to group reference targets in one entry. "Bar. . . 10, *see also* Foo; Baz" is better than "Bar. . . 10, *see also* Foo; *see also* Baz."[18]

\IndexRef causes an index tag with the format ⟨*some text*⟩|⟨*some function*⟩ to be reduced to ⟨*some text*⟩ in the cross-reference. This allows cross-references to work with any index function text used by \TagName (Section 2.7.6).

\ExcludeName
**3.0**

This macro prevents a name from being used as either an index entry or as an index cross-reference. It can be used at any point in the document. It ignores extant cross-references. The syntax is:

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

For example, you will not see any index references for the following names. We show formatting with **boldface**:

```
\ExcludeName[Kris]{Kringle}
\Name[Kris]{Kringle} and \Name[Kris]{Kringle}:
```
**Kris Kringle** and Kringle. \ForgetName will give us a first-use name again:

[17] When the verbose option is selected, \IndexRef warns that a name once used as a page number entry is now being used as a cross-reference. It also warns when one attempts to redefine or alter an established cross-reference.

[18] Professional indexers often use programs like cindex that enforce a rigorous, standard methodology and syntax. The nameauth package likewise tries to follow suit.

**Kris Kringle**, then Kringle.

```
\ExcludeName[Santa]{Claus}
\AKA[Kris]{Kringle}[Santa]{Claus}
```
Santa Claus (not formatted by `\AKA`).

This can be used to prevent references in the index after you are done with a name. Unlike `\IndexInactive` and `\IndexActive` this macro does not suspend the indexing system, but only works on a per-name basis.

`\IncludeName`
`\IncludeName*`
**3.0**

Feel like breaking the indexing rules set by nameauth? Some might want to do things differently. These macros have the same syntax as `\ExcludeName`:

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

The unstarred form of `\IncludeName` only removes the exclusion placed on a page-number entry by `\ExcludeName`. The starred form of `\IncludeName` makes the nameauth macros forget about both an exclusion and a cross-reference.

For example, we used `\ExcludeName{Attila, the Hun}` after his appearance in Section 2.1.2. Using `\IfAKA` from Section 2.9.1 tells us that, "Attila is excluded." Now if we `\IncludeName{Attila, the Hun}`, a reference to Attila will create an index entry on this page. `\IfAKA` now tells us that, "Attila is a name."

In similar fashion, using `\IfAKA` on the cross-reference "Jay Rockefeller" that we created in Section 2.1.2 tells us that "Jay is a cross-reference." If we use the unstarred form `\IncludeName[Jay]{Rockefeller}`, we still get "Jay is a cross-reference." Nothing happened because cross-references are given extra protection. Yet `\IncludeName*[Jay]{Rockefeller}` produces "Jay is a name." Now all protection of "Jay Rockefeller" is removed and we can do anything to it.

### 2.7.5 Index Sorting

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

#### Basic Sorting (Makeindex)

`\PretagName`
**2.0**

The nameauth package integrates this sort of index sorting automatically by using a "pretag." The syntax is:

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

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

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

One need only "pretag" names once in the preamble. Every time that one refers to Jan Łukasiewicz or Æthelred II, the proper index entry will be created. If you create a cross-reference with `\AKA` and you want to "pretag" it, see Section 2.10.

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

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

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

### Extra Spaces and Sorting

Under NFSS, active Unicode characters expand to add one or two spaces after control sequences. See \indexentry and \item entries in your idx and ind files. For example, ä becomes \IeC␣{\"a} (one space added) and Æ becomes \IeC␣{\AE␣} (two spaces added).

Section 2.11.3 shows how this is related to the number of times the active character must be expanded. The character Æ must expand twice, through both \IeC and \T1, while ä expands only once through \IeC to a letter. The character ß (*scharfes Ess, Esszett*) below expands twice.

Both xelatex and lualatex (using fontspec) avoid these issues by handling the characters natively. That produces the following:

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

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

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

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

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

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

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

These are not the only instances of macros inserting extra spaces. If something is off in the index, the best advice is to look at the idx or ind files. You can use the verbatim package to look at the ind file within your job itself:

```
\usepackage{verbatim}
\newif\ifdebug

\ifdebug
  \verbatiminput{\jobname.ind}
\fi
```

### 2.7.6 Index Tags

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

```
\TagName{Leo, I}{, pope}          (in the preamble)
\TagName{Gregory, I}{, pope}
...
\Name*{Leo, I} was known as       Leo I was known as
\AKA{Leo, I}{Leo}[the Great].     Leo the Great.
...
\Name{Gregory, I} ``\ForceFN%     Gregory "the Great,"
\AKA*{Gregory, I}{Gregory}%       another pope.
[the Great],'' another 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 2.1 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 manual is tagged with at least |hyperpage to allow hyperlinks in the index using the ltxdoc class and hypdoc package. You may have to use \string before |hyperpage where a vertical bar is active. For example, the following will produce a special index entry with no page number:

```
\newcommand\orphan[2]{#1}
\TagName{Sine Nomine}{\string|orphan{(Sine Pagina)}}
```

Using \Name{Sine Nomine} Sine Nomine here will generate this entry in the ind file: \item Sine Nomine\pfill \orphan{(Sine Pagina)}{29}. One can use this method to create special index entries via \IndexName, then "protect" the names used with \ExcludeName after the entries have been created.

\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, using macros from Section 2.9.2:

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

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.[19]

## 2.8 "Text Tags"

Section 2.7.6 deals with similar tagging features in the index. "Text tags" differ from index tags because they are not printed automatically with every name managed by nameauth.

These "text tags" are a name information database. The macros in this section are named accordingly. Section 2.11.6 offers additional examples.

\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 "George Washington." Note, however, that the tag does 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)." As with index tags, one can put a space at the start of a tag — or not, depending on the use. Sections 2.7.6 and 2.11.5f. illustrate how this can permit tags like asterisks, daggers, and footnotes, such as one for Schuyler Colfax.[20]

[19] 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.

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

The source for the previous example looks like:

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

By using these text tag macros with the conditional macros, one can display information associated with a name based on whether or the name has occurred. For the example below, we turn indexing off:

```
\NameAddInfo{Sam}
{%
  \IfMainName{Freddy}%
    {\Name{Freddy}'s sidekick}%
    {a young gardener with a Midlands accent}%
}
There is \Name{Sam}. He is \NameQueryInfo{Sam}.
Then \Name{Sam} met \Name{Freddy}, who lives%
with his posh uncle \Name{Bill}.
Now he is \NameQueryInfo{Sam} on a quest to save the realm.
```

There is Sam. He is a young gardener with a Midlands accent.
Then Sam met Freddy, who lives with his posh uncle Bill.
Now he is Freddy's sidekick on a quest to save the realm.

`\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.

## 2.9 Name Decisions

### 2.9.1 Testing Decisions

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

- a book where one ties information to the first mention of a name, like a "text tag," margin paragraph, mini-bio, footnote, etc.
- a game book where you have to pick a game path
- a presentation that can change if certain names are present
- conditional comments using the comment, pdfcomment, and similar packages

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

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

See Sections 2.11.6 and 2.11.7 for related ideas about tokens and expansion. Using `\tracingmacros`, `\show`, or `\meaning` can help you.

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

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

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

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

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

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

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

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

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

This macro works the same as `\IfMainName`. A "front matter" name is 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 2.6, we see that "Carnap is both" a formatted and unformatted name with the following test:

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

Please refer to Sections 2.9.2 and 2.11.2 to understand the scope and operation of main- and front-matter names.

\IfAKA If you want to produce output or perform a task based on whether a cross-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 2.7.4).

A cross-reference name is created by \IndexRef, \AKA, and \AKA*. The following example illustrates how we use this macro:

1. In the text we refer to Jesse Ventura, \Name[Jesse]{Ventura}.
2. We establish his lesser-known legal name as an alias: "James Janos," \AKA[Jesse]{Ventura}[James]{Janos}.
3. We construct the following test:

```
\IfAKA[James]{Janos}%
  {\Name[Jesse]{Ventura} has an alias}%
  {\Name[Jesse]{Ventura} has no alias}%
  {\Name[Jesse]{Ventura} is excluded}
```

4. This gives us "Ventura 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 2.7.4) 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}
```

### 2.9.2 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. Cross-reference names created by `\IndexRef` or `\AKA` are not affected by these macros.

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

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

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

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

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

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

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

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

Scope The default behavior of these two macros changes whether a name is "forgotten" or "subverted" simultaneously for front matter and main matter names, Remember the example on page 32 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 2.6), 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 (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 2.11.2 offers a long example.

[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.

## 2.10 Name Variant Macros

**3.0** The macros in this section are specialized and have a somewhat different syntax than others in this manual. Macros like `\IndexRef` permit one to avoid them altogether. Yet here they are, if needed.

`\AKA` `\AKA*` `\AKA` (meaning *also known as*) handles the occasional full-name mention of pseudonyms, stage names, *noms de plume*, and so on. The syntax for `\AKA` is:

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

Both macros create a cross-reference in the index from the ⟨*Alt. FNN*⟩, ⟨*Alt. SNN*⟩, and ⟨*Alt. names*⟩ fields to a target defined by ⟨*FNN*⟩ and ⟨*SNN*⟩, regardless of whether that name exists. **The name order for `\AKA` is opposite that of `\IndexRef`.** That is due to the following problem:

`[`⟨$FNN_1$⟩`]{`⟨$SNN_1$⟩`}` | `[`⟨$Alt_1$⟩`][`⟨$FNN_2$⟩`]` Which to pick? | `{`⟨$SNN_2$⟩`}[`⟨$Alt_2$⟩`]`

By only allowing ⟨*FNN1*⟩ and ⟨*SNN1*⟩ for the target name, we can let the other fields permit an unrestricted cross-reference. See also Section 2.7.6.

`\AKA` only prints long names in the text. It designed for the occasional mentioning of full alternate names. See page 38 for alternate solutions. `\SeeAlso` works with `\AKA`, `\AKA*`, and `\PName`.

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

**3.0** `\AKA*` prints short name references like `\FName`, meaning that `\ForceFN` works with it in the same manner. For the older behavior of `\AKA*` use the `oldAKA` option or always precede `\AKA*` with `\ForceFN`.

### Basic Operation

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

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

These are the more common alternate-name forms after the main name:

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

These alternate-name forms use non-Western alternate names:

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

These alternate-name forms use the old syntax:

| | | | |
|---|---|---|---|
| `\AKA` | ... | `{`⟨*Alt. SNN*⟩`}[`⟨*Alt. FNN*⟩`]` | *Eastern* |
| `\AKA` | ... | `{`⟨*Alt. SNN*⟩`}[`⟨*Alt. Affix*⟩`]` | *Ancient* |

The next example makes "alternate name" cross-references to the target Bob Hope, illustrating Western names:

| | |
|---|---|
| `\AKA[Bob]{Hope}[Leslie Townes]{Hope}` | Leslie Townes Hope |
| `\AKA*[Bob]{Hope}[Leslie Townes]{Hope}` | Leslie Townes |
| `\AKA[Bob]{Hope}%`<br>`  [Leslie Townes]{Hope}[Leslie]` | Leslie Hope |
| `\AKA*[Bob]{Hope}%`<br>`  [Leslie Townes]{Hope}[Leslie]` | Leslie |

As with nicknames and `\Name`, the alternate forms in the text do not appear in the index. Remember that one cannot apply an index tag to a cross-reference; only a "pretag" will work.

With Louis XIV and Lao-tzu below, notice that there is no change in printed form between `\AKA` and `\AKA*`:

| | |
|---|---|
| `\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 |

This final example illustrates how `\AKA*` can be useful with medieval affixes as we consider Gregory "the Great":

| | |
|---|---|
| `\AKA{Gregory, I}{Gregory}[the Great]` | Gregory the Great |
| `\AKA*{Gregory, I}{Gregory}[the Great]` | Gregory |
| `\ForceFN\AKA*{Gregory, I}%`<br>`  {Gregory}[the Great]` | the Great |

### Formatting Alternate Names: General

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

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

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

`formatAKA`

"Jean the Fearless" usually receives no formatting because it is post-processed by `\MainNamesHook` in the main matter text and `\FrontNamesHook` in the front matter. The `formatAKA` option causes `\AKA` to use `\NamesFormat` and `\FrontNamesFormat`, but **only once in the whole document**. Using the `alwaysformat` option formats all names as first uses. One can tweak the `formatAKA` option with the following and place it before `\AKA` or `\AKA*`:

```
\makeatletter
\newcommand*\Format{\@nameauth@FirstFormattrue}
\makeatother
```

Below we show the effects of `formatAKA` and `alwaysformat` on the example macro `\AKA{Elizabeth, I}[Good Queen]{Bess}`. We use special formatting for main-matter and front-matter names.

`formatAKA`

**Front Matter:** *Elizabeth I* was known as "*Good Queen Bess*." Again we mention Queen Elizabeth, "Good Queen Bess."
`\Format\AKA{Elizabeth, I}[Good Queen]{Bess}` *Good Queen Bess*.
**Main Matter:** Elizabeth I was known as "Good Queen Bess." Again we mention Queen Elizabeth, "Good Queen Bess."
`\Format\AKA{Elizabeth, I}[Good Queen]{Bess}` Good Queen Bess.

`alwaysformat`

**Front Matter:** *Elizabeth I* was known as "*Good Queen Bess*." Again we mention Queen *Elizabeth*, "*Good Queen Bess*."
**Main Matter:** Elizabeth I was known as "Good Queen Bess." Again we mention Queen Elizabeth, "Good Queen Bess."

### Formatting Alternate Names: Continental

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

1. Tag the names for proper sorting.

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

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

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

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

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

4. `\FName` prints "Heinz."

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

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

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

The resulting text is:

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

**3.0** `\AKA` will not create multiple cross-references. Handle the special case where one moniker applies to multiple people with `\IndexRef`, *e.g.*, "Snellius" for both Willebrord Snel van Royen and his son Rudolph Snel van Royen:[22]

```
\IndexRef{Snellius}{Snel van Royen, R.; Snel van Royen, W.}
```

Cross-references generated by `\AKA` and `\AKA*` are meant only to be cross-references, never page entries. See also Section 2.13. In certain cases, the alternate name might need to be indexed with page numbers and *see also* references:

- Refer to the person intended, *e.g.*:
  Maimonides (Moses ben-Maimon):
  `\Name{Maimonides} (\AKA{Maimonides}{Moses ben-Maimon})`
- We now have a target entry and a *see* reference.
- We also should refer to the main name. The fact that we had a target reference does not establish the index entry for the main name. Thus:
  Maimonides
  `\Name{Maimonides}`
- Before creating a cross-reference, one must refer to the alternate name, *e.g.*:
  Rambam
  `\Name{Rambam}`

**3.0**

- For whatever name you use for the *see also* reference, put the cross-reference after all of the page references. For example, you could put both of these macros at the end of the document:[23]
  `\SeeAlso\IndexRef{Maimonides}{Rambam}`
  `\SeeAlso\IndexRef{Rambam}{Maimonides}`
- You could let the last reference to either name be handled by `\SeeAlso\AKA`, but that could be more confusing and prone to error.

Using `\PretagName` (Section 2.7.5) helps to avoid the need for manual index entries. Instead of doing a lot of extra work with `makeindex` for some names, consider the following example:

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

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

We use the medieval form: `\Name{Thomas, Aquinas}` because "Aquinas" is not a surname, even though many people, including scholars, falsely use it as such. Section 2.4.6 talks about those unfortunate situations where one must use the Western form `\Name[Thomas]{Aquinas}`.

[22] We shorten the index entries via `\Name[W.]{Snel van Royen}[Willebrord]`, and for his son, `\Name[R.]{Snel van Royen}[Rudolph]`.

[23] 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.

### General Tips for \AKA

- [⟨*FNN*⟩]{⟨*SNN*⟩} is the target name. [⟨*Alt. FNN*⟩]{⟨*Alt. SNN*⟩}[⟨*Alt. names*⟩] is the cross-reference to the target. Neither create an index entry with page references for the target.
- 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 2.12.
- Reversing works, *e.g.*,
  \RevComma\AKA...: Hope, Leslie Townes
  \RevName\AKA... : Yakumo KOIZUMI

- The name fields of \PretagName correspond with the [⟨*Alt. FNN*⟩]{⟨*Alt. SNN*⟩}[⟨*Alt. names*⟩] fields of \AKA:

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

- With stage names like The Amazing Kreskin, if you want them in the index, use \Name[The Amazing]{Kreskin} to get "Kreskin, The Amazing." Otherwise use something like \Name[J.]{Kreskin}[The Amazing] to get The Amazing Kreskin in the text and "Kreskin, J." in the index.

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

- Special cases like "Iron Mike" Tyson as the nickname for Mike Tyson may be handled in a number of ways.

  – Follow ``Iron Mike'' with \IndexName[Mike]{Tyson} and do whatever you want in the text. This may be the easiest solution.
  – Use ``\AKA[Mike]{Tyson}{Iron Mike}'' to create "Iron Mike" in the text and a cross-reference to "Tyson, Mike" in the index. Be sure to have an occurrence of \Name[Mike]{Tyson} in the text. See also Section 2.10. This is the best solution in terms of how nameauth is designed.
  – Always get "Iron Mike Tyson" with something like:

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

  ``\Iron'' gives you "Iron Mike Tyson." 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.

`\PName` `\PName*`

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

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

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

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

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

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

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

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

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

Also choose forms like `\PName{Lao-tzu}{Li, Er}` "Lao-tzu (Li Er)" instead of `\PName{Lao-tzu}{Li}[Er]` "Lao-tzu (Li Er)." Both forms will work, but the latter form looks confusing and could lead to error.

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

## 2.11 Longer Examples

### 2.11.1 Variant Spellings

Some issues can be a bit tricky to address. Here we show first uses of names in **boldface** to illustrate more clearly what is going on. For example, the problem of using **W.E.B. Du Bois** and the alternate form **W.E.B. DuBois** illustrates a name collision for `nameauth` because the names differ only in terms of spaces.

Normally, that sort of collision helps the fault-tolerant aspects of this package and is a good thing. Here it is not useful. We need to disambiguate the name forms. We do that by inserting an optional hyphen into the alternate form:

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

We still can prevent either name from breaking by putting either `\DuBois` or `\AltDuBois` into an `\hbox`, and so on.

**3.0** Indexing both name forms would be trivial. One can use both at need to generate page references in the index. After all of the page references are done, one can create cross-references with `\SeeAlso\IndexRef`.

Indexing with only the canonical name form Du Bois gets more complex. Before the first reference `\Name[W.E.B.]{Du\-Bois}` or `\AltDuBois` "DuBois" occurs, we prevent the creation of any index entries for it by creating a *see* reference, as we did at the start of this section:

```
\IndexRef[W.E.B.]{Du\-Bois}{Du Bois, W.E.B.}
```

You can see that first and subsequent name forms, as well as formatting, still work. `\ForgetName[W.E.B.]{Du\-Bois}` produces **W.E.B. DuBois**. We simply will not see index references generated for DuBois.

In order to index under `\AltDuBois` as if it were `\DuBois`, we need a macro that will work with the punctuation detection in case we generate the short name W.E.B. We choose the following solution:

```
\global\newcommand*\OtherDuBois
  {\IndexName[W.E.B.]{Du Bois}%
  \AltDuBois\IndexName[W.E.B.]{Du Bois}}
\global\newcommand*\SOtherDuBois
  {\IndexName[W.E.B.]{Du Bois}\SAltDuBois}
```

If we want to "forget" DuBois again via `\ForgetName[W.E.B.]{Du\-Bois}` we get `\OtherDuBois`: **W.E.B. DuBois** and DuBois. With `\SOtherDuBois` we can mention W.E.B. The extra full stop at the end of the sentence was gobbled. We used `\global` to ensure that, regardless of scope, our macros would work wherever we want them to work.

### 2.11.2 \LocalNames

As mentioned previously in Section 2.9.2, both \ForgetName and \SubvertName usually affect both main-matter and front-matter names. This default behavior can be quite helpful. Nevertheless, there are cases where it is undesirable. This section shows \Localnames and \Globalnames in action, limiting the behavior of the "tweaking macros" to either the main or front matter.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

### 2.11.3 Unicode and NFSS

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

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

Some of these characters expand differently, which can affect index sorting. For example, ä becomes `\IeC␣{\"a}` and Æ becomes `\IeC␣{\AE␣}`. Additional accents and glyphs can be used with Unicode input, NFSS, inputenc, and fontenc when using fonts with TS1 glyphs, *e.g.*, `\usepackage{lmodern}` (per the table on pages 455–63 in *The Latex Companion*). The following example lets you type, "In Congreſs, July 4, 1776."

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

Although `\newunicodechar{ā}{\=a}` allows `\Name{Ghazāli}` to generate Ghazāli, one must be careful with control sequences like `\=a`. They fail when using `makeindex` and `gind.ist`. For example, the ltxdoc class, with `gind.ist`, turns the default "actual" character `@` into `=`. Using `\index{Gh{\=a}zali}` halts execution. Using `\index{Gh\=azali}` gives an "azali" entry sorted under "Gh" (thanks Dan Luecking). This issue is not specific to nameauth.

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

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

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

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

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

Without digging into the details of font encoding and NFSS, we can say in simple terms that `æ` is "two arguments wide." Any macro where this `#1#2` pair gets split into `#1` and `#2` will produce either `Unicode char ...not set up for LaTeX` or `Argument of \UTFviii@two@octets has an extra }`. Again, this is not just specific to nameauth.

**3.0** `\CapThis` avoids these pitfalls by checking if the leading token of the argument to be capitalized is equivalent to the leading token of an active Unicode character. We chose `ß` as the test character somewhat at random. Page 63 shows the test. Essentially, the following two expressions are equal under NFSS:

`\@car`⟨*test*$_1$⟩`\@nil`, where ⟨*test*$_1$⟩ expands to `\IeC {`⟨*test*$_1$⟩`}`
`\@car`⟨*test*$_2$⟩`\@nil`, where ⟨*test*$_2$⟩ expands to `\IeC {`⟨*test*$_2$⟩`}`

If ⟨*test*$_2$⟩ expands to the letter ⟨*test*$_2$⟩, then it will fail the equality. "Active" characters expand to "two-argument wide" values under NFSS:

| `\def\a{`⟨*L*⟩`}` | `\protected@edef\a{`⟨*L*⟩`}` | `\protected\edef\a{`⟨*L*⟩`}` |
|---|---|---|
| A macro:->A | A macro:->A | A \protected macro:->A |
| À macro:->ÃĂ | À macro:->\IeC {\‘A} | À \protected macro:->À |
| ß macro:->Ã§ | ß macro:->\IeC {\ss } | ß \protected macro:->\T1\ss |

The number of spaces inserted in the index file depends on the number of expansions that occur for a given active character.

LaTeX also removes spaces between undelimited macro arguments, but not from the trailing undelimited argument. This is no longer an issue for name arguments in nameauth, but we include the information anyway:

| 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> |

Using explicit spacing macros prevents gobbled spaces:

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

See also Sections 2.4.6 and 2.4.7.

### 2.11.4 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:[24]

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

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

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

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

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

### 2.11.5 Hooks: Intro

Margin Paragraphs

Before we get to the use of text tags and name conditionals in name formatting, we begin with an intermediate example to illustrate that something more complex can occur in `\NamesFormat`. Here we put the first mention of a name in boldface, along with a marginal notation if possible:[25]

```
\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} (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" consider using `\Name{Vlad, III}` and use `\NameAddInfo` and `\NameQueryInfo` to handle "`Dracula`." The simplified interface greatly helps one to avoid confusion and settle on specific name forms.

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

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

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

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

### 2.11.6 Hooks: Life Dates

We can use name conditionals (Section 2.9.1) and text tags (Section 2.8) to add life information to names when desired.

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

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

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

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

We assume that we will not be using the alwaysformat option, meaning that we only call this hook once for a first use:[26]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

If we `\let\FrontNamesFormat\NamesFormat` we can get similar results in the front matter and its name system.

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

```
\NameAddInfo{Atatürk}{ (a special surname granted 1934)}
\NameAddInfo{Ata\-türk}{ (a special surname granted 1934)}
```

In order to prevent multiple index entries, I prevented the form `{Ata\-türk}` from appearing in the index via `\IndexInactive` within a scope.

Please remember to reset the formatting, if needed:

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

See Section 3.4 and page 73 for the decision paths and the logic used by the package. Presently, writing hook macros is much simpler.

### 2.11.7 Hooks: Advanced

#### The Name Parser for Hooks

**3.0** The current modular design of nameauth (Section 3.4 and page 73) permits one to predict the state of the macros and Boolean flags in the locked path. This means we can use that available information to "start from scratch" and completely ignore the parsed and formatted argument of the formatting hooks. We do that by having the hook take an argument:

```
\renewcommand*⟨Hook⟩[1]{...}
```

\NameParser There is a pared-down parser available to the package user called \NameParser that only can be used in a formatting hook, else it does nothing. Its sole purpose is to print a name how the regular nameauth parser does it, minus the capitalization extras. Reversing and commas are still usable. The way to use the parser is simple:

```
\renewcommand*⟨Hook⟩[1]
  {⟨cseq1 . . . cseqm⟩ \NameParser ⟨cseqn . . . ⟩}
```

The notation $\langle cseq_1 \ldots cseq_m \rangle$ and $\langle cseq_n \ldots \rangle$ means a number of control sequences that can change the way a name is parsed. The subsequent examples take advantage of this.

Also be aware that if you designed your own hooks for versions of nameauth before 3.0, it remains highly likely that they still work, but without the newer features. Updating your custom hooks is strongly advised.

#### Continental Format

This implementation of Continental formatting prints surnames in small caps the first time they are used, then no small caps thereafter. Yet small caps are used consistently in the index.

The font change from small caps to normal is generated only in the body text output. The way to handle that is by using \noexpand, since the name arguments in nameauth have to use \protected@edef to work right.

In the document preamble, before defining any names, create the following boolean flag and macro:

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

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

\DoFormat is the key to this approach. We want a control sequence that will expand differently, depending on the state of this Boolean flag. Using \noexpand is another vital piece of the solution.

Since Continental formatting alters the surname, see how the formatting macro always appears in the ⟨*SNN*⟩ field below as we start to define names:

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

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

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

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

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

**3.0** We do not need to redefine either `\NamesFormat` or `\FrontNamesFormat` because we already set the default to be small caps. We redefine `\MainNameHook` and `\FrontNameHook` in order to *suppress* formatting in subsequent uses of names. Currently, this is fairly easy.[27]

```
\renewcommand*\MainNameHook[1]{\SCfalse\NameParser}
\let\FrontNameHook\MainNameHook
```

Here we see the results for both main- and front-matter names. Not only did we get the formatting that we wanted, but we also retained many other features.

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

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

- Punctuation detection still works: Sammy Davis Jr. Then we have Davis.
- `\RevComma\LJQA` yields Adams, John Quincy. All the reversing macros work.
- `\DropAffix\LJRIII` gives: John David Rockefeller.
- `\ForceFN\SChas` produces: "I" instead of `\SChas` "Charles." This is, in fact, the correct result.
- Using the main interface requires some extra typing. For example, `\ForgetName[John Quincy]{\noexpand\DoFormat{Adams}}\JQA` results in John Quincy Adams.

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

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

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

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

### Caps within Formatting

This method of redesigning the hooks must be used if you want to capitalize a name that is otherwise formatted in the index. All we need to do is modify the example above. We begin in similar fashion, but add a few extra bits. First come three Boolean flags that control the format changes. `\ifItal` controls italics. `\ifFirstCap` controls capitalization, and `\ifInHook` is set true in a hook macro so that we do not capitalize outside of it.[28]

```
\newif\ifItal%                          Flag to trigger italics
\newif\ifFirstCap%                         Flag to trigger caps
\newif\ifInHook%          But only with a name in the body text
\Italtrue%                         We want italics in the index
```

This `\DoFormat` below is quite similar to the previous example. In fact, if you take the names from the previous example and use them here, this will change the appearance not only in the body text but also in the index, creating a separate set of entries (at least with `makeindex`).

```
\def\DoFormat#1%
{%
  \ifItal \textit{#1}%
  \else #1%
  \fi
}
```

[28] One could use the internal flag `\if@nameauth@InHook` to the same effect. A similar version of this example is in `examples.tex`, collocated with this manual.

Here is the macro that achieves capitalization in the name. Please put its argument within braces to avoid breaking active Unicode characters in NFSS. We have to restrict the operation of this macro to a formatting hook via `\ifInHook`.

```
\def\CP#1%
{%
  \ifInHook
    \ifFirstCap \uppercase{#1}%
    \else #1%
    \fi
  \else #1%
  \fi
}
\newcommand*\CapMe{\FirstCaptrue}
```

Like the previous example, you put the formatting in the naming macro arguments. Remember to ensure that `\CP` does not break active characters by putting its argument in braces.

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

You need to have `\global` before `\FirstCapfalse` to ensure that the change persists beyond the scope of the hook.

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

```
\renewcommand*\NamesFormat[1]
{%
  \InHooktrue\NameParser\InHookfalse%
  \global\FirstCapfalse%
}

\renewcommand*\MainNameHook[1]
{%
  \Italfalse\InHooktrue\NameParser\InHookfalse%
  \global\FirstCapfalse\Italtrue%
}

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

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

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

The capitalized version `\CapMe\deSmet` is De Smet. This also works for a formatted use: *De Smet*. The index entries will be consistent for all the variations in the text.

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

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

### 2.11.8 Full Redesign

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

\NameauthName
\NameauthLName
\NameauthFName

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

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

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

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

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

\global

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

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

## 2.12 Naming Pattern Reference

### Western Names

We set up a `nameauth` environment for each section below.[29]

```
\begin{nameauth}
  \< Einstein & Albert & Einstein & >
  \< Lewis & Clive Staples & Lewis & >
\end{nameauth}
```

| | | |
|---|---|---|
| *First reference in the text*: Albert Einstein | `\Einstein`<br>`\LEinstein`<br>`\SEinstein` | `\Name [Albert]{Einstein}`<br>`\Name*[Albert]{Einstein}`<br>`\FName[Albert]{Einstein}` |
| *Subseq. full*: Albert Einstein | `\LEinstein` | `\Name*[Albert]{Einstein}` |
| *Subsequent surname*: Einstein | `\Einstein` | `\Name [Albert]{Einstein}` |
| *Subsequent forename*: Albert | `\SEinstein` | `\FName[Albert]{Einstein}` |
| *Long first reference*: Clive Staples Lewis | `\Lewis`<br>`\LLewis`<br>`\SLewis` | `\Name [Clive Staples]{Lewis}`<br>`\Name*[Clive Staples]{Lewis}`<br>`\FName[Clive Staples]{Lewis}` |
| *Subsequent full*: C.S. Lewis | `\LLewis[C.S.]` | `\Name*[Clive Staples]{Lewis}[C.S.]` |
| *Subsequent surname*: Lewis | `\Lewis [C.S.]`<br>`\Lewis` | `\Name [Clive Staples]{Lewis}[C.S.]`<br>`\Name [Clive Staples]{Lewis}` |
| *Alternate*: Jack | `\SLewis[Jack]` | `\FName[Clive Staples]{Lewis}[Jack]` |

### Western Plus Affixes

Always use a comma to delimit name/affix pairs.

```
\begin{nameauth}
  \< Patton & George S. & Patton, Jr. & >
\end{nameauth}
```

| | | |
|---|---|---|
| *First reference*: George S. Patton Jr. | `\Patton`<br>`\LPatton`<br>`\SPatton` | `\Name [George S.]{Patton, Jr.}`<br>`\Name*[George S.]{Patton, Jr.}`<br>`\FName[George S.]{Patton, Jr.}` |
| *Subsequent full*: George S. Patton Jr. | `\LPatton` | `\Name*[George S.]{Patton, Jr.}` |
| *Subsequent surname*: Patton | `\Patton` | `\Name [George S.]{Patton, Jr.}` |
| *Subsequent forename*: George S. | `\SPatton` | `\FName[George S.]{Patton, Jr.}` |
| *Alternate forename*: George | `\SPatton[George]`<br>`\FName[George S.]{Patton, Jr.}[George]` | |

[29] For those who observe that `\Lewis` as defined here is different than earlier in this manual, five stars to the boarding school house of your choice. The `nameauth` environment permists one to make successive global redefinitions. Yet this is done at your own risk.

### New Syntax: Non-Western

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

For sobriquets, one could use `\Name{Demetrius, I Soter}` to keep the number with the affix or `\Name{Demetrius I, Soter}` to keep the number with the name. We recommend, however, that you use text tags and index tags for "Soter" (Sections 2.7.6 and 2.8). That is what we do below:

```
\NameAddInfo{Demetrius, I}{Soter}
\def\Soter{\NameQueryInfo{Demetrius, I}}
\TagName{Demetrius, I}{ Soter}
```

| | | |
|---|---|---|
| *First reference*: Francis I | `\Francis` | `\Name {Francis, I}` |
| | `\LFrancis` | `\Name*{Francis, I}` |
| | `\SFrancis` | `\FName{Francis, I}` |
| *Subsequent full*: Francis I | `\LFrancis` | `\Name*{Francis, I}` |
| *Subsequent name*: Francis | `\Francis` | `\Name {Francis, I}` |
| | `\SFrancis` | `\FName{Francis, I}` |
| *First reference*: Demetrius I Soter | `\Dem\ \Soter` | `\Name {Demetrius, I}\ \Soter` |
| | `\LDem\ \Soter` | `\Name*{Demetrius, I}\ \Soter` |
| | `\SDem\ \Soter` | `\FName{Demetrius, I}\ \Soter` |
| *Subsequent full*: Demetrius I | `\LDem` | `\Name*{Demetrius, I}` |
| *Subsequent name*: Demetrius | `\Dem` | `\Name {Demetrius, I}` |
| | `\SDem` | `\FName{Demetrius, I}` |
| *First reference*: Sun Yat-sen | `\Sun` | `\Name {Sun, Yat-sen}` |
| | `\LSun` | `\Name*{Sun, Yat-sen}` |
| | `\SSun` | `\FName{Sun, Yat-sen}` |
| *Subsequent full*: Sun Yat-sen | `\LSun` | `\Name*{Sun, Yat-sen}` |
| *Subsequent name*: Sun | `\Sun` | `\Name {Sun, Yat-sen}` |
| | `\SSun` | `\FName{Sun, Yat-sen}` |
| *Personal name*: Yat-sen | `\ForceFN\SSun` | `\ForceFN\FName{Sun, Yat-sen}` |
| *All references*: Plato | `\Plato` | `\Name {Plato}` |
| | `\LPlato` | `\Name*{Plato}` |
| | `\SPlato` | `\FName{Plato}` |

You also can "stack" `\CapThis`, `\CapName`, `\RevName`, `\KeepAffix`, and so on in front of these control sequences. `\CapName\LSun` generates SUN Yat-sen. For a fuller listing, see Section 2.1.

### Old Syntax: Royal and Eastern

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

For sobriquets, one could use `\Name{Ptolemy, I Soter}` to keep the number with the affix or `\Name{Ptolemy I, Soter}` to keep the number with the name. We recommend, however, that you use text tags and index tags for "Soter" (Sections 2.7.6 and 2.8). That is what we do below:

```
\NameAddInfo{Ptolemy}[I]{Soter}
\def\Soter{\NameQueryInfo{Ptolemy}[I]}
\TagName{Ptolemy}[I]{ Soter}
```

| | | |
|---|---|---|
| *First reference*: Henry VIII‡ | `\Henry` | `\Name {Henry}[VIII]` |
| | `\LHenry` | `\Name*{Henry}[VIII]` |
| | `\SHenry` | `\FName{Henry}[VIII]` |
| *Subsequent full*: Henry VIII‡ | `\LHenry` | `\Name*{Henry}[VIII]` |
| *Subsequent name*: Henry‡ | `\Henry` | `\Name {Henry}[VIII]` |
| | `\SHenry` | `\FName{Henry}[VIII]` |
| *First reference*: Ptolemy I Soter‡ | `\Ptol\ \Soter` | `\Name {Ptolemy}[I]\ \Soter` |
| | `\LPtol\ \Soter` | `\Name*{Ptolemy}[I]\ \Soter` |
| | `\SPtol\ \Soter` | `\FName{Ptolemy}[I]\ \Soter` |
| *Subsequent full*: Ptolemy I‡ | `\LPtol` | `\Name*{Ptolemy}[I]` |
| *Subsequent name*: Ptolemy‡ | `\Ptol` | `\Name {Ptolemy}[I]` |
| | `\SPtol` | `\FName{Ptolemy}[I]` |
| *First reference*: Mao Tse-tung‡ | `\Mao` | `\Name {Mao}[Tse-tung]` |
| | `\LMao` | `\Name*{Mao}[Tse-tung]` |
| | `\SMao` | `\FName{Mao}[Tse-tung]` |
| *Subsequent full*: Mao Tse-tung‡ | `\LMao` | `\Name*{Mao}[Tse-tung]` |
| *Subsequent name*: Mao‡ | `\Mao` | `\Name {Mao}[Tse-tung]` |
| | `\SMao` | `\FName{Mao}[Tse-tung]` |
| *Personal name*: Tse-tung | `\ForceFN\SMao` | `\ForceFN\FName{Mao}[Tse-tung]` |

**3.0** 

There used to be more ambiguity between the new syntax and the old syntax. That has been cleared up in present versions. Now it is clear that the new syntax takes precedence. `\Name*{Æthelred, II}[the Unready]` prints "Æthelred the Unready" in the body text and "Æthelred, II" in the index. In former versions of this package it would have produced "Æthelred, II the Unready" in the text and in the index. **Since the older behavior was discouraged, it no longer will be supported and no backward compatibility will exist.**

Sections 2.8 and 2.7.6, as well as this section, show how one might use tags to handle "the Unready" in a more complex and more consistent manner.

### Particles, American Style

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

| | | |
|---|---|---|
| *First*: Walter de la Mare | `\DLM` | `\Name [Walter]{de la Mare}` |
| | `\LDLM` | `\Name*[Walter]{de la Mare}` |
| | `\SDLM` | `\FName[Walter]{de la Mare}` |
| *Subsequent*: de la Mare | `\DLM` | `\Name [Walter]{de la Mare}` |
| *Start of sentence*: De la Mare | `\CapThis\DLM` | `\CapThis\Name[Walter]{de la Mare}` |
| *Forename*: Walter | `\SDLM` | `\FName[Walter]{de la Mare}` |

### Particles, European Style, in the Index

| | | |
|---|---|---|
| *First use*: Johann Wolfgang von Goethe | `\JWG [Johann Wolfgang von]` | `\Name [J.W. von]{Goethe}% [Johann Wolfgang von]` |
| | `\LJWG[Johann Wolfgang von]` | `\Name*[J.W. von]{Goethe}% [Johann Wolfgang von]` |
| | `\SJWG[Johann Wolfgang von]` | `\FName[J.W. von]{Goethe}% [Johann Wolfgang von]` |
| *Subsequent*: Goethe | `\JWG` | `\Name [J.W. von]{Goethe}` |
| *Forenames*: Johann Wolfgang | `\SJWG[Johann Wolfgang]` | `\FName[J.W. von]{Goethe}% [Johann Wolfgang]` |

### Particles, European Style, Not in the Index

| | | |
|---|---|---|
| *First*: Adolf von Harnack | `\Harnack [Adolf von]` | `\Name [Adolf]{Harnack}[Adolf von]` |
| | `\LHarnack[Adolf von]` | `\Name-[Adolf]{Harnack}[Adolf von]` |
| | `\SHarnack[Adolf von]` | `\FName[Adolf]{Harnack}[Adolf von]` |
| *Subsequent full*: Adolf von Harnack | `\LHarnack[Adolf von]` | `\Name*[Adolf]{Harnack}[Adolf von]` |
| *Subseq. last*: Harnack | `\Harnack` | `\Name [Adolf]{Harnack}` |
| *Subseq. first*: Adolf | `\SHarnack` | `\FName[Adolf]{Harnack}` |

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

## 2.13 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. Format each part separately.
- Consider using `\PretagName` with all names containing control sequences or active Unicode; see Section 2.7.5.
- 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.

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." That is a Western alternate name form, which is incorrect.
- The older syntax will not work with some macros. The comma-suffixed form does work with those macros. See Section 2.10.

Warnings result from the following:

- Using the `nameauth` environment to redefine shorthands or define shorthands that collide with extant macros generates warning because that could result in unwanted behavior like unexpected name forms and index entries. The following will create a warning for such reasons:

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

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

  Sometimes dedefinition is harmless because it produces no unwanted results. It is up to the user to consider these warnings.
- Use the `verbose` option for warnings from the indexing macros.
- Using an index cross-reference name as a page entry. Nothing will happen.
- Creating the same cross-reference multiple times. Nothing will happen.
- Creating a page reference after a cross-reference has been created or after you have used `\ExcludeName`. Nothing will happen until you use a variant of `\Includename`.
- Using `\TagName` and `\UntagName` on cross-references. Nothing will happen.
- Using `\PretagName` with cross-references will create sorting tags for them, but also will generate "informational warnings" only if the `verbose` option is selected.
- Using `\ExcludeName` with cross-references. Nothing will happen.
- Using `\ExcludeName` to exclude a name that has already been excluded. Likewise, it will do nothing.

# 3 Implementation

## 3.1 Flags and Registers

The flags below are grouped according to general function. We begin with flow control

### Who Called Me?

These values are used by the format hook dispatcher `\@nameauth@Hook` and the hook macros to determine if they have been called by either `\@nameauth@Name` or `\AKA`, respectively. Those macros set these flags. On their use, see Sections 2.11.6 and 2.11.7.

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

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

### Core Macro Lock

The macros `\@nameauth@Name` and `\AKA`, with some auxiliary macros, process names in a "locked" state. These flags prevent a stack overflow. See also Sections 2.11.6 and 2.11.7.

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

### Indexing

As the naming macros have locks, so do the indexing macros. These locks permit or prevent both indexing and tags. `\IndexActive` and `\IndexInctive` or the `index` and `noindex` options toggle this flag:

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

The `pretag` and `nopretag` options toggle the value below, which allows or prevents the insertion of sort keys.

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

Theis flag determines whether `\IndexRef` creates a *see* reference or a *see also* reference.

```
\newif\if@nameauth@SeeAlso
```

### Formatting

`\NamesActive` and `\NamesInactive`, with the `mainmatter` and `frontmatter`, options toggle formatting hooks via `\if@nameauth@MainFormat`. `\if@nameauth@AKAFormat` permits `\AKA` to call the first-use hooks once.

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

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

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

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

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

Next we move from general flow control to specific modification of name forms.

## Affix Commas

The `comma` and `nocomma` options toggle the flag value below. `\ShowComma` and `\NoComma` respectively toggle the second and third.

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

## Name Breaking

`\KeepAffix` toggles the flag below, which causes an ⟨*SNN*⟩-⟨*Affix*⟩ pair to have a non-breaking space between them. For "native" Eastern names this prevents the family name and personal name from breaking.

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

## Detect Punctuation

This Boolean value is used to prevent double full stops at the end of a name in the text.

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

## Long and Short Names

`\if@nameauth@FullName` is true in any case where a long name reference is desired. `\if@nameauth@FirstName` disables full-name references and causes only Western forenames to be displayed. `\if@nameauth@AltAKA` is toggled respectively by `\AKA` and `\AKA*` to print a longer or shorter name. `\if@nameauth@OldAKA` forces the pre-3.0 behavior of `\AKA*`. `\if@nameauth@ShortSNN` is used with `\DropAffix` to suppress the affix of a Western name. `\if@nameauth@EastFN` toggles the forced printing of Eastern forenames.

```
\newif\if@nameauth@FullName
\newif\if@nameauth@FirstName
\newif\if@nameauth@AltAKA
\newif\if@nameauth@OldAKA
\newif\if@nameauth@ShortSNN
\newif\if@nameauth@EastFN
```

## Eastern Names

The next flags values govern name reversing and full surname capitalization. The first of each pair is a global state. The second of each pair is an individual state.

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

## Last-Comma-First

This pair of flags deals with Western names reordered in a list according to surname.

```
\newif\if@nameauth@RevAllComma
\newif\if@nameauth@RevThisComma
```

## Capitalize First Letter

The next three flags deal with first-letter capitalization. The first Boolean value is triggered by `\CapThis` and reset by `\Name` and `\AKA`. The second is triggered by `\@nameauth@UTFtest` when it encounters a Unicode character under NFSS. The third is an "override switch" triggered by `\AccentCapThis` as a fall-back in case there is a problem with the detection.

```
\newif\if@nameauth@DoCaps
\newif\if@nameauth@UTF
\newif\if@nameauth@Accent
```

### Warning Levels

This flag controls how many warnings you get. Defaults to few warnings. Verbose gives you plenty of warnings about cross-references in the index.

```
\newif\if@nameauth@Verbose
```

### Name Argument Token Registers

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

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

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

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

## 3.2 Hooks

`\NamesFormat` Post-process "first" instance of final complete name form in text. See Sections 2.6 and 2.11.5f. Called when both `\@nameauth@MainFormat` and `\@nameauth@FirstFormat` are true.

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

`\MainNameHook` Post-process subsequent instance of final complete name form in main-matter text. See Sections 2.6 and 2.11.5f. Called when `\@nameauth@MainFormat` is true and the Boolean flag `\@nameauth@FirstFormat` is false.

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

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

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

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

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

`\NameauthName` Hook to create custom naming macros. Usually the three macros below have the same control sequence, but they need not do so if you want something different. See Section 2.11.8. 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 2.11.8.

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

`\NameauthFName` Customization hook called after `\@nameauth@FirstName` is set true. See Section 2.11.8.

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

## 3.3 Package Options

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

```
\DeclareOption{comma}{\@nameauth@AlwaysCommatrue}
\DeclareOption{nocomma}{\@nameauth@AlwaysCommafalse}
\DeclareOption{mainmatter}{\@nameauth@MainFormattrue}
\DeclareOption{frontmatter}{\@nameauth@MainFormatfalse}
\DeclareOption{formatAKA}{\@nameauth@AKAFormattrue}
\DeclareOption{oldAKA}{\@nameauth@oldAKAtrue}
\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{}}
\DeclareOption{verbose}{\@nameauth@Verbosetrue}
\ExecuteOptions%
  {nocomma,%
   mainmatter,%
   index,%
   pretag,%
   normalcaps,%
   notreversed,%
   noformat}
\ProcessOptions\relax
```

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

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

The etoolbox package is essential for processing name control sequences. Using xargs allows the optional arguments to work. Using suffix facilitated the starred form of macros. Finally, trimspaces helps the fault tolerance of name arguments.

## 3.4 Internal Macros

### Name Control Sequence: Who Am I?

`\@nameauth@Clean` Thanks to Heiko Oberdiek, this macro produces a "sanitized" string used to make a (hopefully) unique control sequence for a name. We can test the existence of that control string to determine first occurrences of a name or cross-reference.

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

## Parsing: Root and Suffix

`\@nameauth@Root` The following two macros return everything before a comma in ⟨*SNN*⟩.

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

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

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

`\@nameauth@TagRoot` The following two macros return everything before a vertical bar (`|`) in an index tag.

```
\newcommand*\@nameauth@TagRoot[1]{\@nameauth@TrimTag#1|\\}
```

`\@nameauth@TrimTag` Throw out the bar and suffix, return the radix.

```
\def\@nameauth@TrimTag#1|#2\\{#1}
```

`\@nameauth@Suffix` The following two macros parse ⟨*SNN*⟩ into a radix and a comma-delimited suffix, returning only the suffix after a comma in the argument, or nothing.

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

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

```
\def\@nameauth@TrimSuffix#1,#2,#3\\{%
  \ifx\\#2\\\@empty\else\trim@spaces{#2}\fi}
```

## Parsing: Capitalization

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

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

`\@nameauth@UTFtest` Before we attempt at capitalizing anything, we need to determine if we are running under `xelatex` or `lualatex` by testing for `\Umathchar`. Then we see if we are un the UTF-8 regime of NFSS. Then we can test if the leading token of our text is the same as the leading token of an active Unicode character. We run this test when we enter `\@nameauth@Name` and `\AKA`.

```
\newcommand*\@nameauth@UTFtest[1]
{%
  \ifdefined\Umathchar
    \@nameauth@UTFfalse%
  \else
    \ifdefined\UTFviii@two@octets
      \toks@\expandafter{\@car#1\@nil}%
      \edef\one{\the\toks@}%
      \toks@\expandafter{\@carß\@nil}%
      \edef\two{\the\toks@}%
      \ifx\one\two\@nameauth@UTFtrue\else\@nameauth@UTFfalse\fi
      \if@nameauth@Accent
        \@nameauth@UTFtrue%
        \@nameauth@Accentfalse%
      \fi
    \else\@nameauth@UTFfalse\fi
  \fi
}
```

\@nameauth@CapRoot Use the NFSS Unicode version if the test is true, else use the regular version.

```
\newcommand*\@nameauth@CapRoot[1]
{%
  \if@nameauth@UTF \@nameauth@CRiii#1\\%
  \else \@nameauth@CRii#1\\%
  \fi
}
```

\@nameauth@CRii Cap the first letter and return the rest as a radix stripped of its suffix.

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

\@nameauth@CRiii Cap the first letter and return the rest as a radix stripped of its suffix. This is called in pdflatex under inputenc/Unicode.

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

## Parsing: 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\\{\TestPunct{##2}}%
  \def\TestPunct##1{%
    \ifx\TestPunct##1\TestPunct%
    \else
      \@nameauth@Puncttrue%
    \fi
  }%
  \@nameauth@Punctfalse%
  \TestDot#1\TestEnd.\TestEnd\\%
}
```

\@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
}
```

## Error Detection

\@nameauth@Error One can cause nameauth to halt with an error by leaving a required name argument empty, providing an argument that expands to empty, or creating a name root that is empty within a root/suffix pair.

```
\newcommand*\@nameauth@Error[2]
{%
  \edef\msga{#2 SNN field empty}%
  \edef\msgb{#2 SNN field malformed}%
  \protected@edef\testname{\trim@spaces{#1}}%
```

```
  \protected@edef\testroot{\@nameauth@Root{#1}}%
  \ifx\testname\@empty
    \PackageError{nameauth}{\msga}%
  \fi
  \ifx\testroot\@empty
      \PackageError{nameauth}{\msgb}%
  \fi
}
```

## Core Name Engine

`\@nameauth@Name` Here is the heart of the package. Marc van Dongen provided the original basic structure. Parsing, indexing, and formatting are more discrete than in earlier versions.

```
\newcommandx*\@nameauth@Name[3][1=\@empty, 3=\@empty]
{%
```

Both `\@nameauth@Name` and `\AKA` engage the lock below, preventing a stack overflow.

```
  \if@nameauth@Lock\else
  \@nameauth@Locktrue%
```

Tell the formatting mechanism that it is being called from `\@nameauth@Name`. Then test for malformed input.

```
  \@nameauth@InNametrue%
  \@nameauth@Error{#2}{macro \string\@nameauth@name}%
```

Ensure that names are printed in horizontal mode. Create an index entry.

```
  \leavevmode\hbox{}%
  \IndexName[#1]{#2}[#3]%
  \if@nameauth@MainFormat
    \@nameauth@Parse[#1]{#2}[#3]{!MN}%
  \else
    \@nameauth@Parse[#1]{#2}[#3]{!NF}%
  \fi
  \IndexName[#1]{#2}[#3]%
```

Reset all the “per name” Boolean values.

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

Close the “locked” branch.

```
  \fi
```

Call the full stop detection.

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

`\@nameauth@Parse` Parse and print a name in the text. The final required argument is a "mode designator" that can be "!MN" (main name); "!NF" (was "non-formatted," now "name in front matter"); and "!PN" (pseudonym/cross-reference). Both `\@nameauth@Name` and `\AKA` call this parser to print their names in the text.

```
\newcommandx*\@nameauth@Parse[4][1=\@empty, 3=\@empty]
{%
  \if@nameauth@Lock
```

We want these arguments to expand to `\@empty` (or not) when we test them.

```
    \protected@edef\arga{\trim@spaces{#1}}%
    \protected@edef\argc{\trim@spaces{#3}}%
    \protected@edef\suffb{\@nameauth@Suffix{#2}}%
```

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
```

Get the root name as a normal or capitalized form, according to the flags.

```
    \if@nameauth@DoCaps
      \@nameauth@UTFtest{#2}%
      \protected@edef\rootb{\@nameauth@CapRoot{#2}}%
    \else
      \if@nameauth@AllThis
        \protected@edef\rootb{\@nameauth@AllCapRoot{#2}}%
      \else
        \protected@edef\rootb{\@nameauth@Root{#2}}%
      \fi
    \fi
```

Make (usually) unique control sequence values from the name arguments.

```
    \def\csb{\@nameauth@Clean{#2}}%
    \def\csbc{\@nameauth@Clean{#2,#3}}%
    \def\csab{\@nameauth@Clean{#1!#2}}%
```

Make token register copies of the current name args to be available for the hook macros.

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

The code below handles non-breaking and regular spaces (`\KeepAffix`). It also handles the use of commas before affixes in Western names.

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

The section below parses names. It "attaches meaning" to the macro arguments via \FNN and \SNN, wich is all the name printing macros know. Call the name printing macros, based on the optional arguments. Then create a second index entry.

```
    \let\SNN\rootb%
    \ifx\arga\@empty
      \ifx\argc\@empty
```

When \arga, \argc, and \suffb are empty, we have a mononym. When \suffb is not empty, we have a native Eastern name or non-Western name.

```
        \let\FNN\suffb%
        \let\SNN\rootb%
        \@nameauth@NonWest{\csb#4}%
      \else
```

When \arga and \suffb are empty, but \argc is not, we have the old syntax. When \arga is empty, but \argc and \suffb are not, we have alternate names for non-Western names.

```
        \ifx\suffb\@empty
          \let\FNN\argc%
          \let\SNN\rootb%
          \@nameauth@NonWest{\csbc#4}%
        \else
          \let\FNN\argc%
          \let\SNN\rootb%
          \@nameauth@NonWest{\csb#4}%
        \fi
      \fi
    \else
```

When \arga is not empty, we have either a Western name or a non-native Eastern name. When \argc is not empty, we use alternate names. When \suffb is not empty we use suffixed forms.

```
      \ifx\argc\@empty
        \let\FNN\arga%
      \else
        \let\FNN\argc%
      \fi
      \ifx\suffb\@empty
      \else
        \protected@edef\SNN{\rootb\Space\suffb}%
        \if@nameauth@ShortSNN\let\SNN\rootb\fi
      \fi
      \@nameauth@West{\csab#4}%
    \fi
  \fi
}
```

`\@nameauth@NonWest` Print non-Western names from `\@nameauth@name` and `\AKA`. We inherit internal control sequences from the naming macros and do nothing if called outside them.

```
\newcommand*\@nameauth@NonWest[1]
{%
  \if@nameauth@Lock
    \ifcsname#1\endcsname\else
      \@nameauth@FirstFormattrue%
    \fi
    \if@nameauth@InAKA
      \if@nameauth@AltAKA
        \if@nameauth@OldAKA\@nameauth@EastFNtrue\fi
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNametrue%
      \else
        \@nameauth@FullNametrue%
        \@nameauth@FirstNamefalse%
      \fi
    \else
      \ifcsname#1\endcsname\else
        \@nameauth@FullNametrue%
        \@nameauth@FirstNamefalse%
      \fi
    \fi
    \if@nameauth@FirstName
      \@nameauth@FullNamefalse%
    \fi
    \ifx\FNN\@empty
      \@nameauth@Hook{\SNN}%
    \else
      \if@nameauth@FullName
        \if@nameauth@RevThis
          \@nameauth@Hook{\FNN\Space\SNN}%
        \else
          \@nameauth@Hook{\SNN\Space\FNN}%
        \fi
      \else
        \if@nameauth@FirstName
          \if@nameauth@EastFN
            \@nameauth@Hook{\FNN}%
          \else
            \@nameauth@Hook{\SNN}%
          \fi
        \else
          \@nameauth@Hook{\SNN}%
        \fi
      \fi
    \fi
    \ifcsname#1\endcsname\else
      \if@nameauth@InAKA\else\csgdef{#1}{}\fi%
    \fi
    \@nameauth@FullNamefalse%
    \@nameauth@FirstNamefalse%
  \fi
}
```

`\@nameauth@West` Print Western names and "non-native" Eastern names from `\@nameauth@name` and `\AKA`. We inherit internal control sequences from the naming macros and do nothing if called outside them.

```
\newcommand*\@nameauth@West[1]
{%
  \if@nameauth@Lock
    \ifcsname#1\endcsname\else
      \@nameauth@FirstFormattrue%
    \fi
    \if@nameauth@InAKA
      \if@nameauth@AltAKA
        \@nameauth@FullNamefalse%
        \@nameauth@FirstNametrue%
      \else
        \@nameauth@FullNametrue%
        \@nameauth@FirstNamefalse%
      \fi
    \else
      \ifcsname#1\endcsname\else
        \@nameauth@FullNametrue%
        \@nameauth@FirstNamefalse%
      \fi
    \fi
    \if@nameauth@FirstName
      \@nameauth@FullNamefalse%
    \fi
    \if@nameauth@FullName
      \if@nameauth@RevThis
        \@nameauth@Hook{\SNN\space\FNN}%
      \else
        \if@nameauth@RevThisComma
          \edef\RevSpace{,\space}%
          \@nameauth@Hook{\SNN\RevSpace\FNN}%
        \else
          \@nameauth@Hook{\FNN\space\SNN}%
        \fi
      \fi
    \else
      \if@nameauth@FirstName
        \@nameauth@Hook{\FNN}%
      \else
        \@nameauth@Hook{\rootb}%
      \fi
    \fi
    \ifcsname#1\endcsname\else
      \if@nameauth@InAKA\else\csgdef{#1}{}\fi%
    \fi
    \@nameauth@FullNamefalse%
    \@nameauth@FirstNamefalse%
  \fi
}
```

### Format Hook Dispatcher

`\@nameauth@Hook` Flags help the dispatcher invoke the correct formatting hooks. The flags control which hook is called (first/subsequent use, name type). The first set of tests handles formatting within `\AKA`. The second set of tests handles regular name formatting. The hooks have a local scope (and are defined as empty) by default.

```
\newcommand*\@nameauth@Hook[1]
{%
  \if@nameauth@Lock
    \@nameauth@InHooktrue%
    \protected@edef\test{#1}%
    \expandafter\@nameauth@TestDot\expandafter{\test}%
    \if@nameauth@InAKA
      \if@nameauth@AlwaysFormat
        \@nameauth@FirstFormattrue%
      \else
        \if@nameauth@AKAFormat\else
        \@nameauth@FirstFormatfalse\fi
      \fi
      \if@nameauth@MainFormat
        \if@nameauth@FirstFormat
          \bgroup\NamesFormat{#1}\egroup%
        \else
          \bgroup\MainNameHook{#1}\egroup%
        \fi
      \else
        \if@nameauth@FirstFormat
          \bgroup\FrontNamesFormat{#1}\egroup%
        \else
          \bgroup\FrontNameHook{#1}\egroup%
        \fi
      \fi
    \else
      \if@nameauth@AlwaysFormat
        \@nameauth@FirstFormattrue%
      \fi
      \if@nameauth@MainFormat
        \if@nameauth@FirstFormat
          \bgroup\NamesFormat{#1}\egroup%
        \else
          \bgroup\MainNameHook{#1}\egroup%
        \fi
      \else
        \if@nameauth@FirstFormat
          \bgroup\FrontNamesFormat{#1}\egroup%
        \else
          \bgroup\FrontNameHook{#1}\egroup%
        \fi
      \fi
    \fi
    \@nameauth@FirstFormatfalse%
    \@nameauth@InHookfalse%
  \fi
}
```

### Indexing Internals

\@nameauth@Index If the indexing flag is true, create an index entry, otherwise do nothing. Add tags automatically if they exist.

```
\newcommand*\@nameauth@Index[2]
{%
  \def\cseq{#1}%
  \let\ex\expandafter%
  \ifcsname\cseq!TAG\endcsname
    \protected@edef\Tag{\csname#1!TAG\endcsname}%
    \ex\def\ex\ShortTag\ex{\ex\@nameauth@TagRoot\ex{\Tag}}%
  \fi
  \if@nameauth@DoIndex
    \ifcsname\cseq!TAG\endcsname
      \ifcsname\cseq!PRE\endcsname
        \if@nameauth@Xref%
          \index%
          {\csname\cseq!PRE\endcsname#2\ShortTag}%
        \else
          \index%
          {\csname\cseq!PRE\endcsname#2\csname\cseq!TAG\endcsname}%
        \fi
      \else
        \if@nameauth@Xref
          \index{#2\ShortTag}%
        \else
          \index{#2\csname\cseq!TAG\endcsname}%
        \fi
      \fi
    \else
      \ifcsname\cseq!PRE\endcsname
        \index{\csname\cseq!PRE\endcsname#2}%
      \else
        \index{#2}%
      \fi
    \fi
  \fi
}
```

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

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

## 3.5 User Interface Macros

### Syntactic Formatting — Capitalization

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

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

\AccentCapThis Overrides the automatic test for active Unicode characters. This is a fall-back in case the automatic test for active Unicode characters fails.

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

\CapName Capitalize entire required 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}
```

`\ForceFN` Force the printing of an Eastern forename in the text, but only when using the "short name" macros.

```
\newcommand*\ForceFN{\@nameauth@EastFNtrue}
```

### Syntactic Formatting — Reversing with Commas

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

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

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

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

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

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

### Syntactic Formatting — Affixes

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

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

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

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

`\DropAffix` Suppress the affix in a long Western name.

```
\newcommand*\DropAffix{\@nameauth@ShortSNNtrue}
```

### Typographic Formatting — Affixes

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

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

### Typographic Formatting — Main Versus Front Matter

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

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

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

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

## Name Occurrence Tweaks

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

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

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

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

## Index Operations

\IndexInactive turn off global indexing of names.

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

\IndexActive turn on global indexing of names.

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

\IndexActual Change the "actual" character from the default.

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

\SeeAlso Change the type of cross-reference from a *see* reference to a *see also* reference. Works once per name.

```
\newcommand*\SeeAlso{\@nameauth@SeeAlsotrue}
```

## Hook Macro Name Parser

\NameParser Generate a name form based on the current state of the nameauth macros in the locked path. Available for use only in the hook macros.

```
\newcommand*\NameParser
{%
  \if@nameauth@InHook
    \let\SNN\rootb%
    \ifx\arga\@empty
      \ifx\argc\@empty
        \let\FNN\suffb%
      \else
        \let\FNN\argc%
      \fi
      \ifx\suffb\@empty
        \SNN%
      \else
        \if@nameauth@FullName
          \if@nameauth@RevThis
            \FNN\Space\SNN%
          \else
            \SNN\Space\FNN%
          \fi
        \else
          \if@nameauth@FirstName
            \if@nameauth@EastFN
```

```
              \FNN%
            \else
              \SNN%
            \fi
          \else
            \SNN%
          \fi
        \fi
      \fi
    \else
      \ifx\argc\@empty
        \let\FNN\arga%
      \else
        \let\FNN\argc%
      \fi
      \ifx\suffb\@empty
      \else
        \protected@edef\SNN{\rootb\Space\suffb}%
        \if@nameauth@ShortSNN\let\SNN\rootb\fi%
      \fi
      \if@nameauth@FullName
        \if@nameauth@RevThis
          \SNN\space\FNN%
        \else
          \if@nameauth@RevThisComma
            \SNN\RevSpace\FNN%
          \else
            \FNN\space\SNN%
          \fi
        \fi
      \else
        \if@nameauth@FirstName
          \FNN%
        \else
          \protected@edef\SNN{\rootb}%
          \SNN%
        \fi
      \fi
    \fi
  \fi
}
```

## Main Naming Interface

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

```
\newcommand\Name{\NameauthName}
```

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

```
\WithSuffix{\newcommand*}\Name*%
  {\@nameauth@FullNametrue\NameauthLName}
```

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

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

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

```
\WithSuffix{\newcommand*}\FName*%
  {\@nameauth@FirstNametrue\NameauthFName}
```

## Index Operations

\IndexName This creates a "main" index entry with page references. It issues warnings if the `verbose` option is selected and if a cross-reference or exclusion exists. It prints nothing. First we make copies of the arguments to test them and make parsing decisions.

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

Test for malformed input.

```
  \@nameauth@Error{#2}{macro \string\IndexName}%
```

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

```
  \let\Short\rootb%
  \ifx\suffb\@empty
    \let\SNN\rootb%
  \else
    \protected@edef\SNN{\rootb\space\suffb}%
  \fi
```

We create the appropriate index entries, calling `\@nameauth@Index` to handle sorting and tagging. We do not create an index entry for a cross-reference (code !PN for pseudonym), used by `\IndexRef`, `\Excludename`, `\Includename`, `\AKA`, and `\PName`.

```
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!PN\endcsname
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \IndexName: XRef: #2 exists}%
        \fi
      \else
        \@nameauth@Index{\csb}{\SNN}%
      \fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \IndexName: XRef: #2 #3 exists}%
          \fi
        \else
          \@nameauth@Index{\csbc}{\SNN\space\argc}%
        \fi
```

```
      \else
        \ifcsname\csb!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \IndexName: XRef: #2 exists}%
          \fi
        \else
          \@nameauth@Index{\csb}{\SNN}%
        \fi
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \IndexName: XRef: #1 #2 exists}%
      \fi
    \else
      \ifx\suffb\@empty
        \@nameauth@Index{\csab}%
        {\Short,\space\arga}%
      \else
        \@nameauth@Index{\csab}%
        {\Short,\space\arga,\space\suffb}%
      \fi
    \fi
  \fi
}
```

`\IndexRef` This creates an index cross-reference that is not already a pseudonym. It prints nothing. First we make copies of the arguments to test them and make parsing decisions.

```
\newcommandx*\IndexRef[4][1=\@empty, 3=\@empty]
{%
  \protected@edef\testa{#1}%
  \protected@edef\arga{\trim@spaces{#1}}%
  \protected@edef\rootb{\@nameauth@Root{#2}}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \protected@edef\testc{#3}%
  \protected@edef\argc{\trim@spaces{#3}}%
  \protected@edef\target{#4}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
  \let\ex\expandafter%
```

Test for malformed input.

```
  \@nameauth@Error{#2}{macro \string\IndexRef}%
  \@nameauth@Xreftrue%
```

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

```
  \let\Short\rootb%
  \ifx\suffb\@empty
    \let\SNN\rootb%
  \else
    \protected@edef\SNN{\rootb\space\suffb}%
  \fi
```

We create either *see also* entries or *see* entries. The former are unrestricted. The latter are only created if they do not already exist as main entries.

```
\ifx\testa\@empty
  \ifx\testc\@empty
    \ifcsname\csb!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \IndexRef: XRef: #2 exists}%
      \fi
    \else
      \if@nameauth@SeeAlso
        \@nameauth@Index{\csb}{\SNN|seealso{\target}}%
      \else
        \@nameauth@Index{\csb}{\SNN|see{\target}}%
      \fi
      \csgdef{\csb!PN}{}%
    \fi
  \else
    \ifx\suffb\@empty
      \ifcsname\csbc!PN\endcsname
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \IndexRef: XRef: #2 #3 exists}%
        \fi
      \else
        \if@nameauth@SeeAlso
          \@nameauth@Index{\csbc}%
            {\SNN\space\argc|seealso{\target}}%
        \else
          \@nameauth@Index{\csbc}%
            {\SNN\space\argc|see{\target}}%
        \fi
        \csgdef{\csbc!PN}{}%
      \fi
    \else
      \ifcsname\csb!PN\endcsname
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \IndexRef: XRef: #2 exists}%
        \fi
      \else
        \if@nameauth@SeeAlso
          \@nameauth@Index{\csb}%
            {\SNN|seealso{\target}}%
        \else
          \@nameauth@Index{\csb}%
            {\SNN|see{\target}}%
        \fi
        \csgdef{\csb!PN}{}%
      \fi
    \fi
  \fi
```

```
  \else
    \ifcsname\csab!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \IndexRef: XRef: #1 #2 exists}%
      \fi
    \else
      \ifx\suffb\@empty
        \if@nameauth@SeeAlso
          \@nameauth@Index{\csab}%
          {\Short,\space\arga|seealso{\target}}%
        \else
          \@nameauth@Index{\csab}%
          {\Short,\space\arga|see{\target}}%
        \fi
      \else
        \if@nameauth@SeeAlso
          \@nameauth@Index{\csab}%
          {\Short,\space\arga,\space\suffb|seealso{\target}}%
        \else
          \@nameauth@Index{\csab}%
          {\Short,\space\arga,\space\suffb|see{\target}}%
        \fi
      \fi
      \csgdef{\csab!PN}{}%
    \fi
  \fi
  \@nameauth@SeeAlsofalse%
  \@nameauth@Xreffalse%
}
```

`\ExcludeName` This macro prevents a name from being indexed.

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

Below we parse the name arguments and create a non-empty pseudonym macro.

```
  \@nameauth@Error{#2}{macro \string\ExcludeName}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \if@nameauth@Verbose
        \ifcsname\csb!MN\endcsname
          \PackageWarning{nameauth}%
          {macro \ExcludeName: Reference: #2 exists}%
        \fi
        \ifcsname\csb!NF\endcsname
          \PackageWarning{nameauth}%
          {macro \ExcludeName: Reference: #2 exists}%
        \fi
      \fi
```

```
      \ifcsname\csb!PN\endcsname
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \ExcludeName: Xref: #2 exists}%
        \fi
      \else
        \csgdef{\csb!PN}{!}%
      \fi
    \else
      \ifx\suffb\@empty
        \if@nameauth@Verbose
          \ifcsname\csbc!MN\endcsname
            \PackageWarning{nameauth}%
            {macro \ExcludeName: Reference: #2 #3 exists}%
          \fi
          \ifcsname\csbc!NF\endcsname
            \PackageWarning{nameauth}%
            {macro \ExcludeName: Reference: #2 #3 exists}%
          \fi
        \fi
        \csgdef{\csbc!PN}{!}%
        \ifcsname\csbc!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \ExcludeName: Xref: #2 exists}%
          \fi
        \else
          \csgdef{\csbc!PN}{!}%
        \fi
      \else
        \if@nameauth@Verbose
          \ifcsname\csb!MN\endcsname
            \PackageWarning{nameauth}%
            {macro \ExcludeName: Reference: #2 exists}%
          \fi
          \ifcsname\csb!NF\endcsname
            \PackageWarning{nameauth}%
            {macro \ExcludeName: Reference: #2 exists}%
          \fi
        \fi
        \ifcsname\csb!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \ExcludeName: Xref: #2 exists}%
          \fi
        \else
          \csgdef{\csb!PN}{!}%
        \fi
      \fi
    \fi
  \else
    \if@nameauth@Verbose
      \ifcsname\csab!MN\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Reference: #1 #2 exists}%
      \fi
```

```
      \ifcsname\csab!NF\endcsname
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Reference: #1 #2 exists}%
      \fi
    \fi
    \ifcsname\csab!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \ExcludeName: Xref: #2 exists}%
      \fi
    \else
      \csgdef{\csab!PN}{!}%
    \fi
  \fi
}
```

`\IncludeName` This macro allows a name to be indexed.

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

Below we parse the name arguments undefine a pseudonym control sequence if it is not excluded (non-empty).

```
  \@nameauth@Error{#2}{macro \string\IncludeName}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!PN\endcsname
        \edef\testex{\csname\csb!PN\endcsname}%
        \ifx\testex\@empty\else\global\csundef{\csb!PN}\fi
      \fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!PN\endcsname
          \edef\testex{\csname\csbc!PN\endcsname}%
          \ifx\testex\@empty\else\global\csundef{\csbc!PN}\fi
        \fi
      \else
        \ifcsname\csb!PN\endcsname
          \edef\testex{\csname\csb!PN\endcsname}%
          \ifx\testex\@empty\else\global\csundef{\csb!PN}\fi
        \fi
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \edef\testex{\csname\csab!PN\endcsname}%
      \ifx\testex\@empty\else\global\csundef{\csab!PN}\fi
    \fi
  \fi
}
```

`\IncludeName*` This macro allows a name to be indexed.

```
\WithSuffix{\newcommandx*}\IncludeName*[3][1=\@empty, 3=\@empty]
{%
  \protected@edef\testa{#1}%
  \protected@edef\testc{#3}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
  \edef\testa{!}%
```

Below we parse the name arguments and undefine a pseudonym control sequence.

```
  \@nameauth@Error{#2}{macro \string\IncludeName*}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \global\csundef{\csb!PN}%
    \else
      \ifx\suffb\@empty
        \global\csundef{\csbc!PN}%
      \else
        \global\csundef{\csb!PN}%
      \fi
    \fi
  \else
    \global\csundef{\csab!PN}%
  \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\testc{#3}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We parse the arguments, defining the sort tag control sequences used by `\@nameauth@Index`.

```
  \@nameauth@Error{#2}{macro \string\PretagName}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!PN\endcsname
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \PretagName: tagging xref: #2}%
        \fi
      \fi
      \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \PretagName: tagging xref: #2 #3}%
          \fi
        \fi
        \if@nameauth@Pretag\csgdef{\csbc!PRE}{#4\@nameauth@Actual}\fi
      \else
        \ifcsname\csb!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \PretagName: tagging xref: #2}%
          \fi
        \fi
        \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \PretagName: tagging xref: #1 #2}%
      \fi
    \fi
    \if@nameauth@Pretag\csgdef{\csab!PRE}{#4\@nameauth@Actual}\fi
  \fi
}
```

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

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

We parse the arguments, defining the index tag control sequences used by `\@nameauth@Index`.

```
  \@nameauth@Error{#2}{macro \string\TagName}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!PN\endcsname
        \if@nameauth@Verbose
          \PackageWarning{nameauth}%
          {macro \TagName: not tagging xref: #2}%
        \fi
      \else
        \csgdef{\csb!TAG}{#4}%
      \fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \TagName: not tagging xref: #2 #3}%
          \fi
        \else
          \csgdef{\csbc!TAG}{#4}%
        \fi
      \else
        \ifcsname\csb!PN\endcsname
          \if@nameauth@Verbose
            \PackageWarning{nameauth}%
            {macro \TagName: not tagging xref: #2}%
          \fi
        \else
          \csgdef{\csb!TAG}{#4}%
        \fi
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \if@nameauth@Verbose
        \PackageWarning{nameauth}%
        {macro \TagName: not tagging xref: #1 #2}%
      \fi
    \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\testc{#3}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We parse the arguments, undefining the index tag control sequences.

```
  \@nameauth@Error{#2}{macro \string\UntagName}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \global\csundef{\csb!TAG}%
    \else
      \ifx\suffb\@empty
        \global\csundef{\csbc!TAG}%
      \else
        \global\csundef{\csb!TAG}%
      \fi
    \fi
  \else
    \global\csundef{\csab!TAG}%
  \fi
}
```

## 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\testc{#3}%
  \protected@edef\Suff{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We parse the arguments, defining the text tag control sequences.

```
  \@nameauth@Error{#2}{macro \string\NameAddInfo}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \csgdef{\csb!DB}{#4}%
    \else
      \ifx\Suff\@empty
        \csgdef{\csbc!DB}{#4}%
      \else
        \csgdef{\csb!DB}{#4}%
      \fi
    \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\testc{#3}%
  \protected@edef\Suff{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We parse the arguments, invoking the tag control sequences to expand to their contents.

```
  \@nameauth@Error{#2}{macro \string\NameQueryInfo}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
    \else
      \ifx\Suff\@empty
        \ifcsname\csbc!DB\endcsname\csname\csbc!DB\endcsname\fi
      \else
        \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
      \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\testc{#3}%
  \protected@edef\Suff{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

We parse the arguments, undefining the text tag control sequences.

```
  \@nameauth@Error{#2}{macro \string\NameClearInfo}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \global\csundef{\csb!DB}%
    \else
      \ifx\Suff\@empty
        \global\csundef{\csbc!DB}%
      \else
        \global\csundef{\csb!DB}%
      \fi
    \fi
  \else
    \global\csundef{\csab!DB}%
  \fi
}
```

## Name Decisions

\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\testc{#3}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

Below we parse the name arguments and choose the path.

```
  \@nameauth@Error{#2}{macro \string\IfMainName}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!MN\endcsname{#4}\else{#5}\fi
      \else
        \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
      \fi
    \fi
  \else
    \ifcsname\csab!MN\endcsname{#4}\else{#5}\fi
  \fi
}
```

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

```
\newcommandx\IfFrontName[5][1=\@empty, 3=\@empty]
{%
  \protected@edef\testa{#1}%
  \protected@edef\testc{#3}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

Below we parse the name arguments and choose the path.

```
  \@nameauth@Error{#2}{macro \string\IfFrontName}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!NF\endcsname{#4}\else{#5}\fi
      \else
        \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
      \fi
    \fi
  \else
    \ifcsname\csab!NF\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\testc{#3}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
```

Fore each class of name we test first if a cross-reference exists, then if it is excluded.

```
  \@nameauth@Error{#2}{macro \string\IfAKA}%
  \ifx\testa\@empty
    \ifx\testc\@empty
      \ifcsname\csb!PN\endcsname
        \edef\testex{\csname\csb!PN\endcsname}%
        \ifx\testex\@empty{#4}\else{#6}\fi
      \else{#5}\fi
    \else
      \ifx\suffb\@empty
        \ifcsname\csbc!PN\endcsname
          \edef\testex{\csname\csbc!PN\endcsname}%
          \ifx\testex\@empty{#4}\else{#6}\fi
        \else{#5}\fi
      \else
        \ifcsname\csb!PN\endcsname
          \edef\testex{\csname\csb!PN\endcsname}%
          \ifx\testex\@empty{#4}\else{#6}\fi
        \else{#5}\fi
      \fi
    \fi
  \else
    \ifcsname\csab!PN\endcsname
      \edef\testex{\csname\csab!PN\endcsname}%
      \ifx\testex\@empty{#4}\else{#6}\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\testc{#3}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
  \def\csb{\@nameauth@Clean{#2}}%
  \def\csbc{\@nameauth@Clean{#2,#3}}%
  \def\csab{\@nameauth@Clean{#1!#2}}%
  \@nameauth@Error{#2}{macro \string\ForgetName}%
```

Now we parse the arguments, undefining the control sequences either by current name type (via `@nameauth@MainFormat`) or completely (toggled by `@nameauth@LocalNames`).

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

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

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

```
\newcommandx*\SubvertName[3][1=\@empty, 3=\@empty]
{%
  \protected@edef\testa{#1}%
  \protected@edef\testc{#3}%
  \protected@edef\suffb{\@nameauth@Suffix{#2}}%
```

```
\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.

```
\@nameauth@Error{#2}{macro \string\SubvertName}%
```

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

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

```
    \fi
  \fi
}
```

## 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%
```

Tell the formatting system that \AKA is running. Test for malformed input.

```
  \@nameauth@InAKAtrue%
  \@nameauth@Error{#2}{macro \string\AKA}%
  \@nameauth@Error{#4}{macro \string\AKA}%
```

Names occur in horizontal mode; we ensure that. Next we make copies of the target name arguments and we parse and print the cross-reference name.

```
  \leavevmode\hbox{}%
  \protected@edef\testi{#1}%
  \protected@edef\argi{\trim@spaces{#1}}%
  \protected@edef\rooti{\@nameauth@Root{#2}}%
  \protected@edef\suffi{\@nameauth@Suffix{#2}}%
  \@nameauth@Parse[#3]{#4}[#5]{!PN}%
```

Create an index cross-reference based on the arguments.

```
  \ifx\testi\@empty
    \ifx\suffi\@empty
      \IndexRef[#3]{#4}[#5]{\rooti}%
    \else
      \IndexRef[#3]{#4}[#5]{\rooti\space\suffi}%
    \fi
  \else
    \ifx\suffi\@empty
      \IndexRef[#3]{#4}[#5]{\rooti,\space\argi}%
    \else
      \IndexRef[#3]{#4}[#5]{\rooti,\space\argi,\space\suffi}%
    \fi
  \fi
```

Reset all the “per name” Boolean values.

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

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{\newcommand*}\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{\newcommand*}\PName*{\@nameauth@FullNametrue\PName}
```

## Simplified Interface

`nameauth` The `nameauth` environment creares macro shorthands.

```
\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\@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}%
    \fi
    \@nameauth@Error{##3}{environment nameauth}%
    \ifcsname\@arga\endcsname
      \PackageWarning{nameauth}%
      {environment nameauth: Shorthand macro already exists}%
   \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\NameauthName%
          \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\@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\ex S\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\ex\ex\ex\ex\@nameauth@FirstNametrue%
          \ex\ex\ex\ex\ex\ex\ex\ex\ex\NameauthFName\ex\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\ex{\ex\the\ex\@nameauth@etoksc\ex}\ex[%
          \the\@nameauth@etoksd]}%
      \fi
    \fi
    \ignorespaces%
  }\ignorespaces%
}{\endgroup\ignorespaces}
```

# 4 Change History

# 5 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.