

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

Charles P. Schaum<sup>†</sup>

Released 2017/01/13

## 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

|          |                                  |           |          |   |            |
|----------|----------------------------------|-----------|----------|---|------------|
| <b>1</b> | <b>Quick Start</b>               | <b>2</b>  | 2.5.4    | Index Sorting . . . . .                           | 32         |
| 1.1      | Introduction . . . . .           | 2         | 2.5.5    | Index Tags . . . . .                              | 34         |
| 1.2      | Basic Concepts . . . . .         | 3         | 2.6      | “Text Tags” . . . . .                             | 35         |
| 1.3      | Traditional Interface . . . . .  | 4         | 2.7      | Name Decisions . . . . .                          | 36         |
| 1.4      | Simplified Interface . . . . .   | 6         | 2.7.1    | Testing Decisions . . . .                         | 36         |
| 1.5      | Older Syntax . . . . .           | 9         | 2.7.2    | Changing Decisions . . .                          | 39         |
| 1.6      | Reference Tables . . . . .       | 10        | 2.8      | Name Variant Macros . . . .                       | 41         |
| <b>2</b> | <b>Detailed Usage</b>            | <b>13</b> | 2.9      | Longer Examples . . . . .                         | 45         |
| 2.1      | Package Options . . . . .        | 13        | 2.9.1    | Variant Names . . . . .                           | 45         |
| 2.2      | Naming Macros . . . . .          | 16        | 2.9.2    | \LocalNames . . . . .                             | 47         |
| 2.2.1    | \Name and \Name* . . . . .       | 16        | 2.9.3    | Unicode and NFSS . . . .                          | 48         |
| 2.2.2    | Forenames: \FName . . . . .      | 17        | 2.9.4    | L <sup>A</sup> T <sub>E</sub> X Engines . . . . . | 50         |
| 2.3      | Language Issues . . . . .        | 18        | 2.9.5    | Hooks: Intro . . . . .                            | 51         |
| 2.3.1    | Affixes Need Commas . . . . .    | 18        | 2.9.6    | Hooks: Life Dates . . . .                         | 52         |
| 2.3.2    | Eastern Names . . . . .          | 19        | 2.9.7    | Hooks: Advanced . . . . .                         | 54         |
| 2.3.3    | Initials . . . . .               | 20        | 2.9.8    | Full Redesign . . . . .                           | 60         |
| 2.3.4    | Hyphenation . . . . .            | 20        | 2.10     | Technical Notes . . . . .                         | 61         |
| 2.3.5    | Listing by Surname . . . . .     | 21        | 2.11     | Errors and Warnings . . . .                       | 62         |
| 2.3.6    | Particles . . . . .              | 21        | <b>3</b> | <b>Implementation</b>                             | <b>63</b>  |
| 2.3.7    | Accented Names . . . . .         | 23        | 3.1      | Flags and Registers . . . . .                     | 63         |
| 2.4      | Formatting . . . . .             | 23        | 3.2      | Hooks . . . . .                                   | 65         |
| 2.4.1    | Spaces & Full Stops . . . . .    | 23        | 3.3      | Package Options . . . . .                         | 66         |
| 2.4.2    | Formatting in the Text . . . . . | 24        | 3.4      | Internal Macros . . . . .                         | 66         |
| 2.4.3    | Alternate Format . . . . .       | 26        | 3.5      | User Interface Macros . . . . .                   | 76         |
| 2.5      | Indexing Macros . . . . .        | 29        | <b>4</b> | <b>Change History</b>                             | <b>99</b>  |
| 2.5.1    | Indexing Control . . . . .       | 29        | <b>5</b> | <b>Index</b>                                      | <b>101</b> |
| 2.5.2    | Index Entries . . . . .          | 30        |          |   |            |
| 2.5.3    | Index Cross-References . . . . . | 31        |          |   |            |

---

\*This file describes version 3.1, last revised 2017/01/13.

<sup>†</sup>E-mail: charles dot schaum at comcast dot net

# 1 Quick Start

## 1.1 Introduction

### 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 (as I know them) are used only for teaching purposes, and I strive to respect those names.

### 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 non-Western syntax are shown with a double dagger (‡). These signs are not added by the package macros and will not appear in users’ works unless they add them.

### Design

When publications use hundreds of names, it takes time and money to manage and check them. This package handles much of that work in order to save time and money. One can implement a name authority, a master list of related names and variants.

- **Automate** 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 name variants only in the body text, not the index.
  - Permit **complex name formatting**. Default is English typography.
  - More **cross-cultural naming conventions** are possible. A basic form of “Continental” formatting has been integrated into the package instead of being a user add-on (Sections 2.3.7, 2.4.3, 2.5.4, and 2.9.7).
  - **Automatic sort keys and tags** aid indexing.
  - One can **automate information retrieval** about names.
  - Indexing generally conforms to the standard in Nancy C. Mulvany, *Indexing Books* (Chicago: University of Chicago Press, 1994). All references [Mulvany] refer to this edition. This was thought suitable for most disciplines.
- 3.0**
- 
- Notable changes correspond to package version numbers in the margin.
  - The “dangerous bend” is used throughout this manual to show where caution is needed to sort out some technical points.
  - Please see Section 2.10 for technical notes regarding general questions about package design, this manual, and the package building and release process.

### 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 users for valuable feedback.

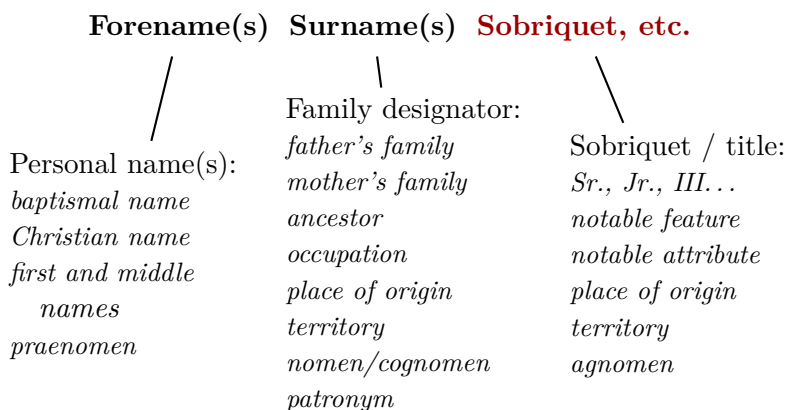
## 1.2 Basic Concepts

Name forms are ambiguous apart from historical and cultural contexts. This package uses that ambiguity to encode names in order to avoid changing the order in which one enters names in one's native culture. In this manual we refer to three general classes of names, shown below. It is helpful to become familiarized with this terminology. Other naming systems can be adapted to these general categories with some caveats, *e.g.*, Icelandic, Hungarian, etc.

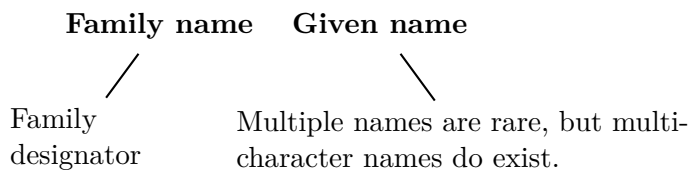
For teaching purposes, we highlight names using sans-serif and use color to show first and subsequent uses of names (see also Sections 2.4.2 and 2.7.2).

Professional writing calls for the full form of a person's name when first used, with shorter forms used thereafter. The name parts that define each class are shown in black, with optional elements in red.<sup>1</sup>

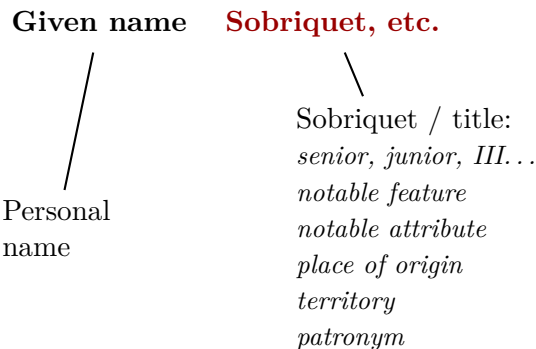
### 1. Western name: George Washington



### 2. Eastern name: Sun Yat-sen



### 3. Ancient name: Elizabeth I




---

<sup>1</sup>Compare [Mulvany, 152–82] and the *Chicago Manual of Style*. That approach is adapted to L<sup>A</sup>T<sub>E</sub>X and its way of handling optional arguments.

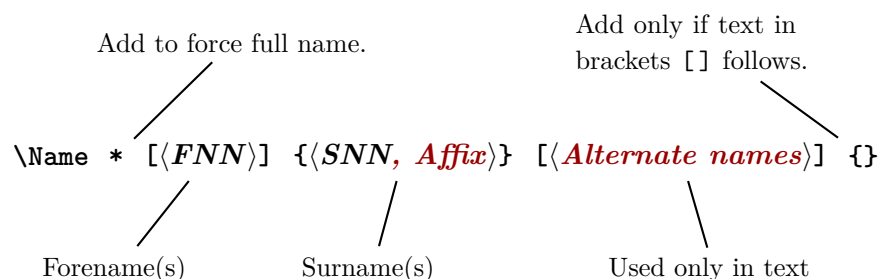
Based on the classes of names, the `nameauth` macros halt with an error in the following cases:

- The required name argument  $\langle SNN \rangle$  expands to the empty string.
- The required argument  $\langle SNN, Affix \rangle$  expands to  $\langle empty \rangle$ ,  $\langle Affix \rangle$ .
- No shorthand is present for a name in the simplified interface (Section 1.4).

### 1.3 Traditional Interface

For all categories, the fields that define each category are shown in black, with optional elements in red.

#### Western Names



#### Examples:

One always must include all fields for consistent index entries.

```
\Name [George]{Washington} ..... George Washington
\Name*[George]{Washington} ..... George Washington
\Name [George]{Washington} ..... Washington
\FName[George]{Washington} ..... George
\Name [George S.]{Patton, Jr.} ..... George S. Patton Jr.
\Name*[George S.]{Patton, Jr.} ..... George S. Patton Jr.
\Name [George S.]{Patton, Jr.} ..... Patton
\FName[George S.]{Patton, Jr.} ..... George S.
```

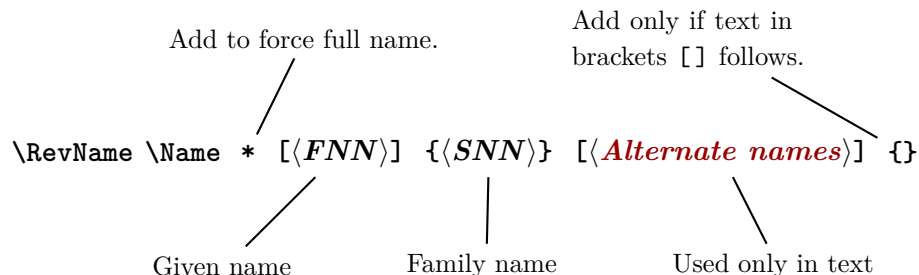
$\langle Alternate names \rangle$  with Western forms require the  $\langle FNN \rangle$  argument to have a name in it.  $\langle Alternate names \rangle$  print in the text.  $\langle FNN \rangle$  prints in the index. For alternate surnames see Section 2.9.1.

```
\Name [Clive Staples]{Lewis} ..... Clive Staples Lewis
\Name*[Clive Staples]{Lewis}[C.S.] ..... C.S. Lewis
\Name [Clive Staples]{Lewis} ..... Lewis
\Name [Clive Staples]{Lewis}[C.S.] ..... Lewis
\Name*[Clive Staples]{Lewis}[Jack] ..... Jack Lewis
\FName[Clive Staples]{Lewis}[Jack] ..... Jack
```

Both affixes and alternate names can vary in the text. Western names require a comma to delimit affixes; see Sections 1.5 and 2.3.1. Using alternate names does not trigger an explicit first use. That is intentional.

```
\Name [John David]{Rockefeller, IV} ..... John David Rockefeller IV
\Name [John David]{Rockefeller, IV}[Jay] ..... Rockefeller
\Name*[John David]{Rockefeller, IV}[Jay] ..... Jay Rockefeller IV
\DropAffix\Name*[John David]{Rockefeller, IV}[Jay] ..... Jay Rockefeller
```

## “Non-Native” Eastern Names in the Text, Western Index Entry



### Examples:

These are encoded using Western name forms without affixes. The reversing macros (Section 2.3.2) cause them to display in Eastern order in the body text. [Mulvany, 166] shows Hungarian names compatible with this category. Index entries are formatted as:  $\langle SNN \rangle$ ,  $\langle FNN \rangle$ . We show these names with a dagger (†).

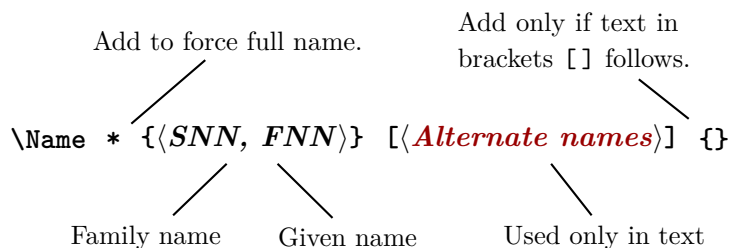
```

\Name[Fumimaro]{Konoe} . . . . . Fumimaro Konoe†
\Name*[Fumimaro]{Konoe}[Prime Minister] . . . . . Prime Minister Konoe†
\RevName\Name*[Fumimaro]{Konoe} . . . . . Konoe Fumimaro†
\RevName\Name[Frenc]{Molnár} . . . . . Molnár Frenc†

```

This “non-native” form of Eastern names excludes both comma-delimited suffixes and the older non-Western syntax (Sections 1.5). This form *will not share* control sequences and index entries with the non-Western forms described below.

## “Native” Eastern Names in the Text, Eastern Index Entry



### Examples:

The main feature of non-Western forms in `nameauth` is the comma-delimited suffix. Eastern names have the family name in  $\langle SNN \rangle$  where ancient names have the personal name, but that remains the required argument.

These names always take the form  $\langle SNN FNN \rangle$  in the index. See Section 2.3.2. Here we call this the “native” Eastern form.

```

\Name{Yamamoto, Isoroku} . . . . . Yamamoto Isoroku
\Name{Yamamoto, Isoroku} . . . . . Yamamoto
\RevName\Name*{Yamamoto, Isoroku} . . . . . Isoroku Yamamoto
\RevName\Name*{Yamamoto, Isoroku}[Admiral] . . . . . Admiral Yamamoto

```

**3.0** Non-Western forms also can have alternate names, except for mononyms (for which it makes no sense). These alternate names do not work with the older syntax for non-Western names (see Section 1.5).

## Ancient Names

Add to force full name.      Name(s)      Add only if text in  
 brackets [] follows.

`\Name * {\SNN, Affix} {}`

### Examples:

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

```

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

## 1.4 Simplified Interface

**nameauth** The **nameauth** environment replaces `\Name`, `\Name*`, and `\FName` with shorthands. Using **nameauth** in the preamble is not required, but it helps prevent undefined control sequences. We set some names up below. Comments are added for explanation; they are not part of the environment itself.

```

%   Field 1   Field 2           Field 3           Field 4
\begin{nameauth}
  \< Wash & George      & Washington      &      > %   Western
  \< Soto  & Hernando    & de Soto      &      > %   Western
  \< Pat   & George S.    & Patton, Jr.  &      > %   W+affix
  \< JRIV  & John David    & Rockefeller, IV &      > %   W+affix
  \< Lewis & Clive Staples    & Lewis        &      > %   Western
  \< Aris  &                    & Aristotle     &      > %   Ancient
  \< Aeth  &                    & Ethelred, II  &      > %   Ancient
  \< Eliz  &                    & Elizabeth, I  &      > %   Ancient
  \< Attil &                    & Attila, the Hun &      > %   Ancient
  \< Konoe & Fumimaro          & Konoe         &      > %   Was East.
  \< Miyaz &                    & Miyazaki, Hayao &      > %   Eastern
  \< Yamt  &                    & Yamamoto, Isoroku &      > %   Eastern
\end{nameauth}
  
```

- Field 1 contains text that will be turned into three control sequences. For example, `Wash` generates `\Wash` (like `\Name`): *George Washington*, `\LWash` (L for Long; like `\Name*`): *George Washington*, and `\SWash` (S for Short; like `\FName`): *George*.
- Fields 2 and 3 hold the name arguments.
- Field 4 usually remains empty. It handles the older non-Western syntax (Section 1.5) and permanent alternate names (next page).
- 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.

- Extra spaces in each &-delimited field are stripped, as is also the case in the traditional interface (Section 2.4.1).
- Put trailing braces {} or something else after the shorthands to prevent subsequent text in brackets [] from becoming an optional argument.

Normally you would use something like \LLewis[C.S.] to get **C.S. Lewis** instead of **Clive Staples Lewis**. You can make that permanent, where **C.S.** always prints in the text, yet the index always shows “Lewis, Clive Staples. Some permanent alternate names are shown below:

```
\begin{nameauth}
  \< JayR & John David & Rockefeller, IV & Jay      > %  Western
  \< CSL  & Clive Staples & Lewis                & C.S.    > %  Western
  \< Unraed & & Æthelred, II                    & Unrædig > %  Ancient
  \< MSens  & & Miyazaki, Hayao                  & Sensei  > %  Eastern
\end{nameauth}
```

So you get **Jay Rockefeller IV**, **C.S. Lewis**, **Æthelred Unrædig**, and **Miyazaki Sensei** instead of **John David Rockefeller IV**, **Clive Staples Lewis**, **Æthelred II**, and **Miyazaki Hayao**, but they all have the same respective index entries.<sup>2</sup> The caveat is that \LLewis[Jack] prints **Jack Lewis** while \LCSL[Jack] prints **C.S. Lewis**[Jack]. Section 2.2.2 explains why.

```
\begin{nameauth}
  \< JWG    & J.W. von & Goethe      & > %      Western; German
  \< VBuren & Martin   & Van Buren & > %      Western; English
\end{nameauth}
```

English keeps the prefix with the surname in the text and the index: **Martin Van Buren** is “Van Buren, Martin” in the index. German separates particles: \JWG prints **J.W. von Goethe** and **Goethe**, with “Goethe, J.W. von” in the index. You get **von Goethe** with \LJWG[von]. See Section 2.3.6 on multicultural standards and Section 2.9.1 on indexing alternate names. Additionally, [Mulvany, 152–82] and the *Chicago Manual of Style* offer helpful guidance.

### So, why use it?

The simplified interface can save work. Instead of the traditional interface macros on the left, one uses the simplified macros on the right:

```
\Name [George]{Washington} ..... \Wash: George Washington
\Name*[George]{Washington} ..... \LWash: George Washington
\Name [George]{Washington} ..... \Wash: Washington
\FName[George]{Washington} ..... \SWash: George

\IndexName[George]{Washington} ..... \JustIndex\Wash
\ForgetName[George]{Washington}%
\Name[George]{Washington} ..... \ForgetThis\Wash: George Washington
\SubvertName[George]{Washington}%
\Name[George]{Washington} ..... \SubvertThis\Wash: Washington
```

<sup>2</sup>One could use \AKA to create a cross-reference **Jay Rockefeller**. See Sections 2.5.3 and 2.8.

## Examples:

Below, “non-native” Eastern name forms are shown with a dagger (†). Please see Section 2.3.2 to avoid pitfalls with Eastern names and reversing macros. We reset some “first uses” of names from before (Section 2.7.2).

|                                      |   |
|--------------------------------------|---|
| WESTERN:                             | ANCIENT / MONONYM                           |
| \Wash ..... George Washington        | \Aris ..... Aristotle                       |
| \LWash ..... George Washington       | \Aris ..... Aristotle                       |
| \Wash ..... Washington               |   |
| \SWash ..... George                  | MEDIEVAL/ROYAL:                             |
| \RevComma\LWash Washington, George   | \Eliz ..... Elizabeth I                     |
|                                      | \Eliz ..... Elizabeth                       |
| PARTICLES: (Section 2.3.6)           | \LEliz[the First] ..... Elizabeth the First |
| \Soto ..... Hernando de Soto         | \Attil ..... Attila the Hun                 |
| \Soto ..... de Soto                  | \Attil ..... Attila                         |
| \CapThis\Soto ..... De Soto          |   |
|                                      | “NON-NATIVE” EASTERN:                       |
| AFFIXES: (Section 2.3.1)             | \Konoe ..... Fumimaro Konoe†                |
| \Pat ..... George S. Patton Jr.      | \LKonoe ..... Fumimaro Konoe†               |
| \LPat ..... George S. Patton Jr.     | \LKonoe[Minister] ..... Minister Konoe†     |
| \DropAffix\LPat .. George S. Patton  | \Konoe ..... Konoe†                         |
| \Pat ..... Patton                    | \SKonoe ..... Fumimaro†                     |
| \SPat ..... George S.                | \CapName\RevName\LKonoe KONOE Fumimaro†     |
|                                      | \CapName\Konoe ..... KONOE†                 |
| NICKNAMES: (Section 2.2.2)           |   |
| \JRIV ... John David Rockefeller IV  | “NATIVE” EASTERN:                           |
| \DropAffix\LRIV[Jay] Jay Rockefeller | \CapName\Yamt ..... YAMAMOTO Isoroku        |
| \SJRV[Jay] ..... Jay                 | \CapName\LYamt ..... YAMAMOTO Isoroku       |
| \Lewis ..... Clive Staples Lewis     | \CapName\Yamt ..... YAMAMOTO                |
| \LLewis[Jack] ..... Jack Lewis       | \RevName\LYamt ..... Isoroku Yamamoto       |
| \SLewis[Jack] ..... Jack             | \RevName\LYamt[Admiral] Admiral Yamamoto    |
| \LCSL ..... C.S. Lewis               | \SYamt ..... Yamamoto                       |
| \SCSL ..... C.S.                     | \ForceFN\SYamt ..... Isoroku                |



Sections 2.3.6, 2.3.7, and 2.5.4 deal with the pitfalls of accents and capitalization, as well as why you should use \PretagName when dealing with names that contain control sequences or active Unicode characters.



The simplified interface can tempt one into completely equating a name with its shortcut. Here we show that to be false. \ForgetThis\CSL prints C.S. Lewis. Then \Lewis prints Lewis. Likewise, \ForgetThis\Lewis prints Clive Staples Lewis. Then \CSL prints Lewis. The name itself is the pattern that governs everything. Internally, that detokenized pattern is CliveStaples!Lewis. Non-western names have patterns like Elizabeth,I and Yamamoto,Isoroku. Mononyms are their own pattern: Aristotle.



For the same reasons, when index tagging or pre-tagging names, the <Alternate names> field has no effect on index tags. \JRIV and \JayR need only one tag, as do \Lewis and \CSL:

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



## 1.5 Older Syntax


An older syntax for non-Western names remains for backward compatibility with early versions of `nameauth`. The older syntax prevents the use of alternate names, limits the use of `\AKA` (Section 2.8) and excludes comma-delimited suffixes. Otherwise it works seamlessly with the new syntax.

The big change is, instead of using a comma-delimited affix, this form uses the final optional argument for personal names and affixes. When `nameauth` was young, this seemed the intuitive approach to take. Now it only remains so that older documents still work today.


```
\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  $\langle FNN \rangle$  fields are empty, the final field becomes either  $\langle affix \rangle$  or  $\langle FNN \rangle$  and will appear in the index. We show these names with a double dagger ( $\ddagger$ ):

|                                      |                            |
|--------------------------------------|----------------------------|
| <code>\Name{Henry}[VIII]</code>      | Henry VIII $\ddagger$      |
| <code>\Name{Henry}[VIII]</code>      | Henry $\ddagger$           |
| <code>\Name{Chiang}[Kai-shek]</code> | Chiang Kai-shek $\ddagger$ |
| <code>\Name{Chiang}[Kai-shek]</code> | Chiang $\ddagger$          |
| <code>\Dagb</code>                   | Dagobert I $\ddagger$      |
| <code>\Dagb</code>                   | Dagobert $\ddagger$        |
| <code>\CapName\Yosh</code>           | YOSHIDA Shigeru $\ddagger$ |
| <code>\CapName\RevName\LYosh</code>  | Shigeru YOSHIDA $\ddagger$ |

- 2.6**  `\Name{Henry}[VIII]` (older syntax) will share name occurrences, tags, and index entries with `\Name{Henry, VIII}` (new syntax), as we see below. We recommend using the newer syntax unless otherwise needed.

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

- 3.0**  Presently `\Name*{Henry, VIII}[Tudor]` prints “Henry Tudor” in the body text and “Henry VIII” in the index. Before version 3.0 it would have produced “Henry VIII Tudor” in the text and in the index. **The older behavior was discouraged. It is obsolete and not supported.** See also Sections 2.5.5 and 2.6.

## 1.6 Reference Tables

### Getting Things Done

Here we link from general tasks to relevant sections. The end of each section listed in the table has a return link to this section.

| <b>I want to...</b>  | <b>Topic</b>        | <b>Section</b>        |
|--|---------------------|-----------------------|
| implement standard scholarly names   | <code>\Name</code>  | <a href="#">2.2.1</a> |
| refer to forenames and affixes   | <code>\FName</code> | <a href="#">2.2.2</a> |
|  | forcing references  | <a href="#">2.7.2</a> |
| use surnames with inflected or alternate forms without creating unwanted index entries | indexing control    | <a href="#">2.5.1</a> |
|  | forcing references  | <a href="#">2.7.2</a> |
|  | alternate spellings | <a href="#">2.9.1</a> |
| use affixes in names   | comma delimiter     | <a href="#">2.3.1</a> |
| use “native” Eastern name forms  | comma delimiter     | <a href="#">2.3.1</a> |
|  | Eastern names       | <a href="#">2.3.2</a> |
| use reversing and all caps for all Eastern name forms in body text only                | Eastern names       | <a href="#">2.3.2</a> |
| use discretionary caps in body text only   | particles           | <a href="#">2.3.6</a> |
| use discretionary caps in text and index   | advanced hooks      | <a href="#">2.9.7</a> |
| handle non-English names and Continental formatting                                    | particles           | <a href="#">2.3.6</a> |
|  | accents             | <a href="#">2.3.7</a> |
|  | non-English format  | <a href="#">2.4.3</a> |
|  | indexing control    | <a href="#">2.5.1</a> |
|  | index sorting       | <a href="#">2.5.4</a> |
|  | advanced hooks      | <a href="#">2.9.7</a> |
| not have affixes be present by default in long name forms                              | index tags          | <a href="#">2.5.5</a> |
|  | text tags           | <a href="#">2.6</a>   |
| manage index cross-references  | cross-references    | <a href="#">2.5.3</a> |
|  | alternate names     | <a href="#">2.8</a>   |
| format variant name forms  | formatting          | <a href="#">2.4.2</a> |
|  | indexing control    | <a href="#">2.5.1</a> |
|  | forcing references  | <a href="#">2.7.2</a> |
|  | alternate spellings | <a href="#">2.9.1</a> |
| use <code>nameauth</code> with beamer overlays   | formatting          | <a href="#">2.4.2</a> |
| or design a game book  | index tags          | <a href="#">2.5.5</a> |
| or design a history book   | text tags           | <a href="#">2.6</a>   |
| or use many dynamic name elements  | name tests          | <a href="#">2.7.1</a> |
| or force name elements to be constant  | forcing references  | <a href="#">2.7.2</a> |
|  | life dates          | <a href="#">2.9.6</a> |
|  | advanced hooks      | <a href="#">2.9.7</a> |

## Form and Format Overview

Below we see how the naming macros generate output. First uses of a name are full references and call first-use formatting hooks. Subsequent uses can be longer or shorter, calling their own hooks unless `\ForceName` changes that (Section 2.4.2). Section 2.7.2 also has more information on how to change things. For changes to `\AKA` and friends, the `alwaysformat` option may be needed (Section 2.8).

### `\Name` or Unmodified Shorthand

| First Reference  | Full | Short | <code>\NamesFormat</code><br><code>\FrontNamesFormat</code> | <code>\MainNameHook</code><br><code>\FrontNameHook</code> |
|------------------|------|-------|---|---|
|                  | Yes  | No    | Yes   | No  |
| Subsequent Ref.  |      |       | §Cf. <code>\ForceName</code>                                |   |
| *Western Surname | No   | *Yes  | §No   | Yes   |
| *Eastern Surname | No   | *Yes  | §No   | Yes   |
| *Ancient Name    | No   | *Yes  | §No   | Yes   |

### `\Name*` or L-modifier + Shorthand

| First Reference | Full | Short | <code>\NamesFormat</code><br><code>\FrontNamesFormat</code> | <code>\MainNameHook</code><br><code>\FrontNameHook</code> |
|-----------------|------|-------|---|---|
|                 | Yes  | No    | Yes   | No  |
| Subsequent Ref. | Yes  | No    | §Cf. <code>\ForceName</code>                                |   |
|                 |      |       | §No   | Yes   |

### `\FName` or S-modifier + Shorthand

| First Reference   | Full | Short | <code>\NamesFormat</code><br><code>\FrontNamesFormat</code> | <code>\MainNameHook</code><br><code>\FrontNameHook</code> |
|-------------------|------|-------|---|---|
|                   | Yes  | No    | Yes   | No  |
| Subsequent Ref.   |      |       | §Cf. <code>\ForceName</code>                                |   |
| *Western Forename | No   | *Yes  | §No   | Yes   |
| *Eastern Surname  | No   | *Yes  | §No   | Yes   |
| *Ancient Name     | No   | *Yes  | §No   | Yes   |

### `\ForceFN\FName` or `\ForceFN` S-modifier + Shorthand

| First Reference   | Full | Short | <code>\NamesFormat</code><br><code>\FrontNamesFormat</code> | <code>\MainNameHook</code><br><code>\FrontNameHook</code> |
|-------------------|------|-------|---|---|
|                   | Yes  | No    | Yes   | No  |
| Subsequent Ref.   |      |       | §Cf. <code>\ForceName</code>                                |   |
| *Western Forename | No   | *Yes  | §No   | Yes   |
| *Eastern Forename | No   | *Yes  | §No   | Yes   |
| *Ancient Affix    | No   | *Yes  | §No   | Yes   |

## Selected Macro Patterns:

|                                 |                             |                              |                               |  |
|---------------------------------|-----------------------------|------------------------------|-------------------------------|--|
| $\langle prefix macros \rangle$ | <code>\Name</code>          | $\langle optional * \rangle$ | $\langle arguments \rangle$   |  |
| $\langle prefix macros \rangle$ | <code>\FName</code>         | $\langle optional * \rangle$ | $\langle arguments \rangle$   |  |
| $\langle prefix macros \rangle$ | <code>\AKA</code>           | $\langle optional * \rangle$ | $\langle target args \rangle$ | $\langle xref args \rangle$                  |
| <code>\SeeAlso</code>           | <code>\IndexName</code>     |                              | $\langle arguments \rangle$   |  |
| <code>\SeeAlso</code>           | <code>\IndexRef</code>      |                              | $\langle arguments \rangle$   | $\langle target \rangle$                     |
|                                 | <code>\ExcludeName</code>   |                              | $\langle arguments \rangle$   |  |
|                                 | <code>\IncludeName</code>   | $\langle optional * \rangle$ | $\langle arguments \rangle$   |  |
|                                 | <code>\PretagName</code>    |                              | $\langle arguments \rangle$   | $\langle sort key \rangle$                   |
|                                 | <code>\TagName</code>       |                              | $\langle arguments \rangle$   | $\langle tag \rangle$                        |
|                                 | <code>\UntagName</code>     |                              | $\langle arguments \rangle$   |  |
|                                 | <code>\NameAddInfo</code>   |                              | $\langle arguments \rangle$   | $\langle tag \rangle$                        |
|                                 | <code>\NameQueryInfo</code> |                              | $\langle arguments \rangle$   |  |
|                                 | <code>\NameClearInfo</code> |                              | $\langle arguments \rangle$   |  |
|                                 | <code>\IfMainName</code>    |                              | $\langle arguments \rangle$   | $\{\langle y \rangle\}\{\langle n \rangle\}$ |
|                                 | <code>\IfFrontName</code>   |                              | $\langle arguments \rangle$   | $\{\langle y \rangle\}\{\langle n \rangle\}$ |
|                                 | <code>\IfAKA</code>         |                              | $\langle arguments \rangle$   | $\{\langle y \rangle\}\{\langle n \rangle\}$ |
|                                 | <code>\ForgetName</code>    |                              | $\langle arguments \rangle$   |  |
|                                 | <code>\SubvertName</code>   |                              | $\langle arguments \rangle$   |  |

## Prefix Macros (One-Time Effect):

They stack: `\CapThis\SubvertThis\SkipIndex\Name[foo]{bar}`: **Bar**

|                           |   |
|---------------------------|---|
| <code>\CapThis</code>     | Capitalize first letter of $\langle SNN \rangle$ or $\langle Affix \rangle$ in body text. Overrides both <code>\CapName</code> and <code>\AllCapsActive</code> . <sup>3</sup> |
| <code>\CapName</code>     | Cap entire $\langle SNN \rangle$ in body text.  |
| <code>\RevName</code>     | Reverse name order in body text (e.g., for Eastern names).  |
| <code>\RevComma</code>    | Reverse Western names to $\langle SNN \rangle$ , $\langle FNN \rangle$ . <sup>4</sup>   |
| <code>\ShowComma</code>   | Add comma between $\langle SNN \rangle$ and $\langle Affix \rangle$ .   |
| <code>\NoComma</code>     | No comma between $\langle SNN \rangle$ and $\langle Affix \rangle$ . Excludes <code>\ShowComma</code> .   |
| <code>\ForceFN</code>     | Force Eastern ForeName or ancient FiNal affix. <sup>5</sup>   |
| <code>\DropAffix</code>   | Drop name affix of Western name (in long name reference). <sup>6</sup>  |
| <code>\KeepAffix</code>   | Insert non-breaking space between $\langle SNN \rangle$ and $\langle Affix \rangle$ . <sup>7</sup>  |
| <code>\KeepName</code>    | Insert non-breaking space between all syntactic name elements.  |
| <code>\ForceName</code>   | Have a subsequent name use call first-use formatting hooks.   |
| <code>\ForgetThis</code>  | Next naming macro prints a first use. Excludes <code>\SubvertThis</code> .  |
| <code>\SubvertThis</code> | The next naming macro prints a subsequent use.  |
| <code>\SeeAlso</code>     | The next cross-reference macro creates a <i>see also</i> reference. <sup>8</sup>  |
| <code>\SkipIndex</code>   | The next naming macro does not create index entries.  |
| <code>\JustIndex</code>   | The next <code>\Name</code> or <code>\FName</code> acts just like a call to <code>\IndexName</code> . Ignored and reset by <code>\AKA</code> and <code>\PName</code> .        |

<sup>3</sup>`\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.

<sup>4</sup>Has no effect on non-Western name forms.

<sup>5</sup>Only affects non-Western name forms.

<sup>6</sup>Only affects Western name forms.

<sup>7</sup>Used best with Western and ancient name forms.

<sup>8</sup>Works only with `\IndexRef`, `\AKA`, `\PName` and their respective starred variants.

## 2 Detailed Usage

### 2.1 Package Options

One includes the `nameauth` package thus:

```
\usepackage[\langle option_1 \rangle,\langle option_2 \rangle,...] {nameauth}
```

The options have no required order. Still, we discuss them from the general to the specific, as the headings below indicate. In the listings below, when there are defaults, **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

|                     |  |
|---------------------|--|
| <b>mainmatter</b>   | <b>Start with “main-matter names” and formatting hooks (see also page 15).</b>                               |
| <b>frontmatter</b>  | Start with “front-matter names” and hooks.   |
| <b>alwaysformat</b> | Use only respective “first use” formatting hooks.  |
| <b>formatAKA</b>    | Format the first use of a name with <code>\AKA</code> like the first use of a name with <code>\Name</code> . |
| <b>oldAKA</b>       | Force <code>\AKA*</code> to act like it did before v.3.0.  |

The `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 2.4.2.

The `alwaysformat` option forces “first use” hooks globally in both naming systems. Its use is limited in current versions of `nameauth`.

- 3.1** The `formatAKA` option permits `\AKA` to use the “first use” formatting hooks. This enables `\ForceName` to trigger those hooks at will (Section 2.8). Otherwise `\AKA` uses “subsequent use” hooks.

- 3.0** Using the `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 2.2.2 and 2.8).

#### Enable/Disable Indexing

|                |  |
|----------------|--|
| <b>index</b>   | <b>Create index entries in place with names.</b> |
| <b>noindex</b> | 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. **Caution:** using `noindex` and `\IndexInactive` prevents index tags until you call `\IndexInactive`, as explained also in Section 2.5.1.



## Enable/Disable Index Sorting

|                 |   |
|-----------------|---|
| <b>pretag</b>   | Create sort keys used with <b>makeindex</b> . |
| <b>nopretag</b> | Do not create sort keys.                      |

The default allows `\PretagName` to create sort keys used with NFSS or `makeindex` and its analogues. The `nopretag` option disables the sorting mechanism, e.g., if a different sorting method is used with `xindy`. See Section 2.5.4.

## Affect the Syntax of Names

### Show/Hide Affix Commas

|                |   |
|----------------|---|
| <b>nocomma</b> | Suppress commas between surnames and affixes, following the <i>Chicago Manual of Style</i> and other conventions. |
| <b>comma</b>   | 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 2.3.1 shows how one can use `\ShowComma` with the `nocomma` option and `\NoComma` with the `comma` option to get per-name results.

### Capitalize Entire Surnames

|                   |  |
|-------------------|--|
| <b>normalcaps</b> | Do not perform any special capitalization.                   |
| <b>allcaps</b>    | 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 2.4.3 and 2.9.7. The simplified interface aids the embedding of control sequences in names. Section 2.3.2 deals with capitalization on a section-level and per-name basis.

### Reverse Name Order

|                    |  |
|--------------------|--|
| <b>notreversed</b> | Print names in the order specified by <code>\Name</code> and the other macros. |
| <b>allreversed</b> | Print all name forms in “smart” reverse order.                                 |
| <b>allrevcomma</b> | Print all names in “Surname, Forenames” order, meant for Western names.        |

These options are mutually exclusive. Section 2.3.2 speaks more about reversing. The `allreversed` option, `\ReverseActive`, and `\RevName` work with all names and override `allrevcomma` and its macros.

So-called “last-comma-first” lists of names via `allrevcomma` and the reversing macros `\ReverseCommaActive` and `\RevComma` (Section 2.3.5) are *not* the same as the `comma` option. They only affect Western names.

## Typographic Post-Processing

### Formatting Attributes

|                        |  |
|------------------------|--|
| <code>smallcaps</code> | First use of a main-matter name in small caps. |
| <code>italic</code>    | First use of a main-matter name in italic.     |
| <code>boldface</code>  | First use of a main-matter name in boldface.   |
| <code>noformat</code>  | <b>Do not define a default format.</b>         |

**2.5** Current versions assign no default formatting to names. Most users have preferred the `noformat` option as the default and then design their own hooks as needed.<sup>9</sup> The options above are “quick” solutions based on English typography.

**2.4** What was “typographic formatting” has become a generalized concept of “post-processing” via hook macros.<sup>10</sup> Post-processing does not affect the index. Sections 2.4.2, 2.9.5, 2.9.6, and 2.9.7 explain these hooks in greater detail:


- `\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.

`\global` Changes to the formatting hooks apply within the scope where they are made. Use `\global` explicitly to alter that. `\NamesFormat` originally was the only hook, so any oddity in the naming of these hooks results from the need for backward compatibility with old versions.

## Alternate or Continental Formatting

### Alternate Syntactic Formatting

|                        |   |
|------------------------|---|
| <code>altformat</code> | Make available the alternate formatting framework from the start of the document. Activate formatting by default. |
|------------------------|---|

**3.1**  A built-in framework provides an alternate formatting mechanism that can be used for “Continental” formatting that one sees in German, French, and so on. Continental standards format surnames only, both in the text and in the index. Section 2.4.3 introduces the topic, while Section 2.9.7 goes into greater detail. The previous methods the produced Continental formatting still ought to work. The error protection that prevents `\CapThis` from breaking alternately formatted names is available by using this option or other macros in Section 2.4.3.

---

<sup>9</sup>For those that want the old default option from the early days of this package, one can recover that behavior with the `smallcaps` option.

<sup>10</sup>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. 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.

## 2.2 Naming Macros

Although the formatting hooks do nothing by default, we use them here for teaching purposes. We also force first and subsequent uses as needed. See also Sections 2.4.2 and 2.7.2. Section 2.7.2 show how the name reference systems are independent of other data sets in nameauth.

### 2.2.1 \Name and \Name\*

`\Name` `\Name` displays and indexes names. It always prints the required “surname” field.  
`\Name*` `\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 older non-Western syntax is used (Section 1.5).

```
\begin{nameauth}  

  < Einstein & Albert & Einstein & >  

  < Cicero & M.T. & Cicero & >  

  < Confucius & & Confucius & >  

  < Miyaz & & Miyazaki, Hayao & >  

  < Eliz & & Elizabeth, I & >  

\end{nameauth}
```

|   |                       |
|---|-----------------------|
| <code>\Name [Albert]{Einstein} or \Einstein</code>  | Albert Einstein       |
| <code>\Name*[Albert]{Einstein} or \LEinstein</code> | Albert Einstein       |
| <code>\Name [Albert]{Einstein} or \Einstein</code>  | Einstein              |
| <code>\Name [M.T.]{Cicero} or \Cicero</code>        | M.T. Cicero           |
| <code>\Name*[M.T.]{Cicero}[Marcus Tullius]</code>   | Marcus Tullius Cicero |
| <code>\Name [M.T.]{Cicero} or \Cicero</code>        | Cicero                |
| <code>\Name {Confucius} or \Confucius</code>        | Confucius             |
| <code>\Name {Miyazaki, Hayao} or \Miyaz</code>      | Miyazaki Hayao        |
| <code>\Name*{Miyazaki, Hayao}[Sensei]</code>        | Miyazaki Sensei       |
| <code>\Name {Miyazaki, Hayao} or \Miyaz</code>      | Miyazaki              |
| <code>\Name {Elizabeth, I} or \Eliz</code>          | Elizabeth I           |
| <code>\Name*{Elizabeth, I} or \LEliz</code>         | Elizabeth I           |
| <code>\Name {Elizabeth, I} or \Eliz</code>          | Elizabeth             |

When using the simplified interface, the preferred way to get alternate names is `\LCicero[Marcus Tullius]` and `\LMiyaz[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 short name references. `\Name[M.T.]{Cicero}[Marcus Tullius]` shows up as just Cicero in that case. By default, subsequent name references are surnames only.

Back to Section 1.6



### 2.2.2 Forenames: \FName

`\FName` `\FName` and its synonym `\FName*` print personal names only in subsequent name uses. They print full names for first 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⟩]`

`\ForceFN` These macros work with both Eastern and Western names, but to get an Eastern personal name, one must precede these macros with `\ForceFN`.<sup>11</sup> See also Section 2.7.2 on how to vary some of the forms below. The standard results for subsequent name uses below are:

**3.0**

|   |                |
|---|----------------|
| <code>\FName[Albert]{Einstein}</code> or <code>\SEinstein</code>                                  | Albert         |
| <code>\FName[M.T.]{Cicero}[Marcus Tullius]</code><br>or <code>\SCicero[Marcus Tullius]</code>     | Marcus Tullius |
| <code>\FName{Confucius}</code> or <code>\SConfucius</code>  | Confucius      |
| <code>\FName{Miyazaki, Hayao}</code> or <code>\SMiyaz</code>                                      | Miyazaki       |
| <code>\ForceFN\FName{Miyazaki, Hayao}</code><br>or <code>\ForceFN\SMiyaz</code>                   | Hayao          |
| <code>\ForceFN\FName{Miyazaki, Hayao}[Sensei]</code><br>or <code>\ForceFN\SMiyaz[Sensei]</code>   | Sensei         |
| <code>\FName{Elizabeth, I}</code> or <code>\SEliz</code>  | Elizabeth      |
| <code>\ForceFN\FName{Elizabeth, I}[the First]</code><br>or <code>\ForceFN\SEliz[the First]</code> | the First      |

The `⟨Alternate names⟩` argument always replaces the forenames in the text. Sometimes this is a good thing, and sometimes it is not:

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

For example, if a book section refers always to **C.S. Lewis**, but another section introduces him as **Clive Staples Lewis**, one can use both `\CSL` and `\Lewis`. `\Lewis` and `\CSL` share common first and subsequent uses because they both point to the same `⟨FNN⟩` (Clive Staples) and `⟨SNN⟩` (Lewis).

The drawback lies in remembering that `\Ches` gives us **Chesley B. Sullenberger III**, while `\LSully` produces **Sully Sullenberger III**. Likewise, `\SCSL[Jack]` produces **C.S.[Jack]**. This happens because the final field in the `nameauth` environment populates the `⟨Alternate Names⟩` argument. When that occurs, the following text in square brackets becomes just normal text.

Back to Section 1.6

<sup>11</sup>Otherwise you would get poor results with some royal and ancient names.

## 2.3 Language Issues

Here we engage topics that relate to specific aspects of grammar and cultural standards. The `nameauth` package is designed with a keen awareness of cross-cultural use and tries to implement such aspects in a smooth fashion.

### 2.3.1 Affixes Need Commas

Comma-delimited affixes are shown below. For Western names, they separate a surname and an affix. For non-Western names, they separate either a surname and a forename or a name and an affix. *Always use a comma as an affix delimiter*, even when commas are not printed. Spaces between the comma and affix are ignored. See also Section 2.4.1.

|  |                      |
|--|----------------------|
| <code>\Name[Oskar]{Hammerstein, II}</code> | Oskar Hammerstein II |
| <code>\Name[Oskar]{Hammerstein, II}</code> | Hammerstein          |
| <code>\Name{Louis, XIV}</code>             | Louis XIV            |
| <code>\Name{Louis, XIV}</code>             | Louis                |
| <code>\Name{Sun, Yat-sen}</code>           | Sun Yat-sen          |
| <code>\Name{Sun, Yat-sen}</code>           | Sun                  |



Western names with suffixes must use the comma-delimited syntax. Using the older non-Western syntax `\Name[Oskar]{Hammerstein}[II]` produces **II Hammerstein** (index entry skipped). Also, one must use comma-delimited suffixes to with the cross-reference target of `\AKA` (Section 2.8).

`\KeepAffix` In the text only, `\KeepAffix` puts a non-breaking space between a Western surname and affix, an ancient name and affix, and a native Eastern family name and personal name. In the text only, `\KeepName` turns all spaces between syntactic name components ( $\langle FNN \rangle$ ,  $\langle SNN \rangle$ ,  $\langle Affix \rangle$ ,  $\langle Alternate\ name(s) \rangle$ ) into non-breaking spaces. You get no bad breaks when using `\KeepName\LJWG[von] von Goethe`. Any spaces between multiple names in each syntactic name component are not affected. `\KeepAffix` and `\KeepName` affect all `nameauth` macros that print a name in the text.

`\DropAffix` 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.

`\ShowComma` `\ShowComma` forces a comma between a Western name and its affix. It works like the `comma` option on a per-name basis, and only in the body text. `\NoComma` works like the `nocomma` option on a per-name basis.

|   |                     |
|---|---------------------|
| <code>\ShowComma\Name*[Louis]{Gossett, Jr.}</code>          | Louis Gossett, Jr.  |
| <code>\NoComma\Name*[Louis]{Gossett, Jr.}</code>            | Louis Gossett Jr.   |
| <code>\RevComma\ShowComma\Name*[Louis]{Gossett, Jr.}</code> | Gossett, Jr., Louis |
| <code>\RevComma\NoComma\Name*[Louis]{Gossett, Jr.}</code>   | Gossett Jr., Louis  |

Back to Section 1.6

### 2.3.2 Eastern Names

non-native The `nameauth` package offers “non-native” and “native” ways to handle romanized Eastern names. The “non-native” form is entered as a Western name and it is indexed as such. `\RevName` reverses its order in the body text:

`\RevName\Name*[\langle FNN \rangle]{\langle SNN \rangle}[\langle Alternate names \rangle]`

The index entry of this name form looks like  $\langle SNN \rangle$ ,  $\langle FNN \rangle$  (including the comma). This type of entry is a Western form. Pick this form especially when using Hungarian names, although apologies are due from this author for needing to enter these names in reverse.

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. Apologies are due for using the  $\langle SNN \rangle$  and  $\langle FNN \rangle$  nomenclature for Eastern family and personal names, but human speech has its limitations. The new syntax permits alternate names; the old does not:

`\Name{\langle SNN, FNN \rangle}[\langle Alternate names \rangle]` (new syntax)  
`\Name{\langle SNN \rangle}[\langle FNN \rangle]` (older syntax)

The index entry of this name form looks like  $\langle SNN \rangle$   $\langle FNN \rangle$  (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` In addition to the class options (Section 2.1), the macros `\ReverseActive`  
`\ReverseInactive` and `\ReverseInactive` toggle reversing on a larger scale, while `\RevName` is used  
`\RevName` 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 (†):

|                                      | <i>unchanged</i>                       | <code>\RevName</code>                  |
|--------------------------------------|--|--|
| <code>\LKonoe</code>                 | Fumimaro Konoe†                        | Konoe Fumimaro†                        |
| <code>\LKonoe[Prime Minister]</code> | Prime Minister Konoe†                  | <i>\langle not appropriate \rangle</i> |
| <code>\Konoe</code>                  | Konoe†                                 | Konoe†                                 |
| <code>\SKonoe</code>                 | Fumimaro†                              | Fumimaro†                              |
| <code>\LYamt</code>                  | Yamamoto Isoroku                       | Isoroku Yamamoto                       |
| <code>\LYamt[Admiral]</code>         | <i>\langle not appropriate \rangle</i> | Admiral Yamamoto                       |
| <code>\Yamt</code>                   | Yamamoto                               | Yamamoto                               |
| <code>\SYamt</code>                  | Yamamoto                               | Yamamoto                               |
| <code>\ForceFN\SYamt</code>          | Isoroku                                | Isoroku                                |

3.0 Creating “last-comma-first” listings by surname (Section 2.3.5) 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 comma 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`      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. Older-syntax forms have a double dagger(‡):

|                                   | <code>\CapName</code> only | <code>\CapName\RevName</code> |
|-----------------------------------|----------------------------|-------------------------------|
| <code>\Name*[Yoko]{Kanno}</code>  | Yoko KANNO†                | KANNO Yoko†                   |
| <code>\Name*{Arai, Akino}</code>  | ARAI Akino                 | Akino ARAI                    |
| <code>\Name*{Ishida}[Yoko]</code> | ISHIDA Yoko‡               | Yoko ISHIDA‡                  |
| <code>\Name*{Yohko}</code>        | 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 2.4.3 and 2.9.7.

Back to Section 1.6

### 2.3.3 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}
```

|                     |             |
|---------------------|-------------|
| <code>\White</code> | E. B. White |
| Normal text:        | E. B. White |

### 2.3.4 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 can make the solution more elegant at times. We avoid bad breaks by using `babel`:

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

### 2.3.5 Listing by Surname

`\ReverseCommaActive` The 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 do not affect “native” Eastern and ancient name forms. Also, see below how long uses are not always first uses:

|                      |                       |           |
|----------------------|-----------------------|-----------|
| Martin Van Buren     | Van Buren, Martin     | OK        |
| Oskar Hammerstein II | Hammerstein II, Oskar | OK        |
| Æthelred II          | Æthelred II           | no change |
| Chiang Kai-shek      | Chiang Kai-shek       | 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:

|                                |                  |
|--------------------------------|------------------|
| <code>\ForgetThis\Konoe</code> | Fumimaro Konoe†  |
| <code>\RevName\LKonoe</code>   | Konoe Fumimaro†  |
| <code>\RevComma\LKonoe</code>  | Konoe, Fumimaro† |

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

### 2.3.6 Particles

According to [Mulvany, 165f.] and 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*. To Anglicize Goethe in the text as *von Goethe*, but indexed under “Goethe, J.W. von,” we use `\LJWG[von]`. `\Name[Catherine de’]{Medici}` should be indexed as “Medici, Catherine de’” instead of modern “De Medici.” See also Sections 2.5.1 and especially 2.9.1 for name variants.

non-breaking We recommend inserting `~` or `\nobreakspace` between particles and names if spaces the particles are less than three letters. This will keep them from becoming lost because of a bad line break or page break.<sup>12</sup> Please remember that some particles look very similar. For example, *L’* and *d’* are two separate glyphs each, while *L* and *d* are one Unicode glyph each.

`\CapThis` In English and modern Romance languages, *e.g.*, *Hernando de Soto* shows that these particles go in the  $\langle SNN \rangle$  field of `\Name`: *de Soto*. When the particle appears at the beginning of a sentence, one must capitalize it:

`\CapThis\Soto\` was a famous Spanish explorer in North America.  
*De Soto* was a famous Spanish explorer in North America.

**3.1** `\CapThis` will not cause errors if one uses the `altformat` option and the provided macros for Continental surname formats. Otherwise it could cause errors in some cases where control sequences in the macro arguments conflict with the capitalization process. See Section 2.4.3.

<sup>12</sup>With v.3.0, `\CapThis` does not eat the space between a single-letter particle and a name.

`\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 67. 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 2.9.3.

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, new syntax
  < KempAlt & & Thomas   & à~Kempis   >      medieval, older syntax
  < KempW    & Thomas   & à~Kempis & >      Western
\end{nameauth}
```

3.1 The medieval forms are **Thomas à Kempis** and **Thomas**, indexed as “Thomas à Kempis.” The suffixed place name “à Kempis” (Latin for *von Kempen*) is used by some as a surname: **à Kempis**. We use `\ForceFN\SKempMed` to get that form. **À Kempis** can start a sentence via `\CapThis\ForceFN\SKempMed`. The old syntax works just as well: **À Kempis** occurs via `\CapThis\ForceFN\SKempAlt`.



Western forms like `\KempW`: **Thomas à Kempis** are different from medieval forms and create different index entries. `\CapThis\KempW` gives “**À Kempis**” in the text and “à Kempis, Thomas” in the index, which we suppress here.<sup>13</sup> The publisher’s way of handling names may differ from the standard way. This package allows for such variations.<sup>14</sup> Developing a good rapport with the publisher and the editor will help you apply this package to the company’s style.

Alternates Using `\CapThis` with a form like `\SubvertThis\ForceFN\FName{Thomas, \‘a~Kempis}` only works in NFSS: **À Kempis**. It fails with `xelatex` and `lualatex` because `inputenc` is not compatible with `fontspec`.



Non-English contexts do not necessarily bind particles to surnames. One can use the alternate names fields or “text tags” and “index tags.” See also Sections 2.5.5, 2.6, and 2.9.6. The macros below allow us to show **Friedrich I Barbarossa**, **Friedrich I**, and **Friedrich** via `\Name{Friedrich, I}`:

```
\NameAddInfo{Friedrich, I}{Barbarossa}
\PretagName{Friedrich, I}{Friedrich I}
\TagName{Friedrich, I}{ Barbarossa, emperor|hyperpage}

\makeatletter\renewcommand*\NamesFormat[1]{\begingroup%
\protected@edef\temp{\endgroup{\color{naviolet}\sffamily #1 %
\noexpand\NameQueryInfo[\unexpanded\expandafter{\the\@nameauth@toksa}]
{\unexpanded\expandafter{\the\@nameauth@toksb}}
{\unexpanded\expandafter{\the\@nameauth@toksc}}}}\temp}\makeatother
```

Back to Section 1.6

<sup>13</sup>Name variants result from work flow constraints, name authorities, and publisher styles. This package works with that, over against this author’s plea for cultural sensitivity.

<sup>14</sup>Yet some publishers have problems with some name forms. 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 a helpful reference work.

### 2.3.7 Accented Names

For names that contain accented characters, using `xelatex` or `lualatex` with `xindy` (`texindy`) is recommended. See also Section 2.9.4.



In NFSS, many Unicode characters are active. Especially with `makeindex`, use `\PretagName` to sort all names with active characters (Sections 2.5.4 and 2.9.3). These active characters differ from explicit control sequences that one might type. We suppress unwanted index entries below among the names that truly are different, yet look the same.

- `\Name{Æthelred, II}` creates `Æthelred II` and `Æthelred`. Now we have a different name: `\Name{\AE thelred, II}` `Æthelred II` (a “first reference”).
- `\Name{Bo\"ethius}` `Boëthius` is not the same as `\Name{Boëthius}` `Boëthius`. Both differ from `\Name{\textsf{Boëthius}}` `Boëthius`.

See Section 2.9.3 on how to add additional Unicode glyphs to the default set under NFSS, `inputenc`, and `fontenc`.

Back to Section 1.6

## 2.4 Formatting

### 2.4.1 Spaces & Full Stops

The `nameauth` package is forgiving with spaces; extra spaces usually do not create unique names, as we see below:

| <i>Macro Example</i>  | <i>Resulting Text</i>               |
|---|-------------------------------------|
| <code>\Name*[Martin Luther]{King, Jr.}</code>   | <code>Martin Luther King Jr.</code> |
| <code>\Name*[<u>  </u>Martin<u>  </u>Luther<u>  </u>]{<u>  </u>King,<u>  </u>Jr.<u>  </u>}</code> | <code>Martin Luther King Jr.</code> |

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 and produce two of them. `\Name`, `\FName`, and `\AKA` detect this in the printed form of a name and gobble the subsequent full stop as needed:

| <i>Macro Example</i>                                     | <i>Result</i> | <i>Resulting Text</i>               |
|--|---------------|-------------------------------------|
| <code>\Name [Martin Luther]{King, Jr.}.</code>           | gobbled       | <code>Martin Luther King Jr.</code> |
| <code>\Name [Martin Luther]{King, Jr.}.</code>           | stayed        | <code>King.</code>                  |
| <code>\Name*[Martin Luther]{King, Jr.}.</code>           | gobbled       | <code>Martin Luther King Jr.</code> |
| <code>\DropAffix\Name*[Martin Luther]{King, Jr.}.</code> | stayed        | <code>Martin Luther King.</code>    |
| <code>\FName[Martin Luther]{King, Jr.}[M.L.].</code>     | gobbled       | <code>M.L.</code>                   |

Grouping tokens can prevent this: `{\Name*[Martin Luther]{King, Jr.}}` produces “`Martin Luther King Jr.`” We see two periods. Enclosing `{Jr.}` within braces or making the whole suffix a macro argument will do the same. Leave the final period outside the macro or group:

```
\Name[Martin Luther]{\textSC{King}, \textSC{Jr}.}
```

Compare Sections 2.4.3 and 2.9.7.



## 2.4.2 Formatting in the Text

There are two kinds of formatting at work that interact with each other:

1. **Syntactic Formatting:** Displayed name elements, reversing, and caps normally occur only in the body text, not the index.
2. **Name Post-Processing:** Hook macros apply formatting to the printed form of a name, which normally does not affect the name form.

`\NamesFormat` Independent “main-matter” and “front-matter” systems format first and  
`\FrontNamesFormat` subsequent name uses. The main-matter system uses `\NamesFormat` to post-  
`\MainNameHook` process first occurrences of names and `\MainNameHook` for subsequent uses.  
`\FrontNameHook` The front-matter system uses `\FrontNamesFormat` for first occurrences and  
**2.5** `\FrontNameHook` for subsequent uses. The `alwaysformat` option causes only  
`\NamesFormat` and `\FrontNamesFormat` to be used. Section 2.7.2 show how the  
name reference systems are independent of other data sets in `nameauth`.

`\NamesActive` `\NamesInactive` and the `frontmatter` option make names use the front mat-  
`\NamesInactive` ter system. `\NamesActive` switches names to the main matter system.  
`\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.

These two systems differ only with respect to first and subsequent name uses. We show this here by using different colors. At the start of this manual, we set up the following after defining our custom colors:

```
\renewcommand*\FrontNamesFormat[1]{\color{naagreen}\sffamily #1}  
\renewcommand*\FrontNameHook[1]{\color{naolive}\sffamily #1}  
\renewcommand*\NamesFormat[1]{\color{naviolet}\sffamily #1}  
\renewcommand*\MainNameHook[1]{\color{naorange}\sffamily #1}
```

The two systems are meant to be used in distinct parts of the document, such as front matter and main matter or text and footnotes. The look awkward when used in the same block of text.

We switch to the “front matter” system:

```
\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” system:

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



`\ForceName` Use this prefix macro to force “first use” formatting for the next `\Name`, etc.  
**3.1** This will not force a full name reference. One must use the `formatAKA` option when using this with `\AKA`, etc. We show this macro in Sections 2.7.2, 2.8, and 2.9.6.


`alwaysformat` 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**; **Elizabeth I**, then **Elizabeth**.

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

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

The hooks are called in a way that lets them either have one argument or none and keeps changes local via: `\bgroup\Hook\{#1}\egroup`

applied to  
 footnotes  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 % restore this later
\long\def\@makefntext#1{%
  \renewcommand*\NamesFormat{\color{naviolet}\scshape}%
  \@oldfntext{#1}}
\makeatother
```

The problem above is that **John Maynard Keynes** in the text affects formatting in the footnotes.<sup>15</sup> In this case, the subsequent-use hook `\MainNameHook` is called because the name already occurred in the text.

Using the front-matter system to manage names in the footnotes independently of those in the body text may be the easiest and most robust solution:

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

Your footnotes do not affect the main body text now.<sup>16</sup> Nevertheless, you can synchronize the two naming systems if needed or manipulate them independently; see Section 2.7.2. Now we change footnotes back to normal, either with `\let` or with scoping and groups:

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

Back to Section 1.6

<sup>15</sup>You get **Keynes** from `\Name[John Maynard]{Keynes}` instead of **JOHN MAYNARD KEYNES**.

<sup>16</sup>We have the expected **John Maynard Keynes**, then **Keynes**.

### 2.4.3 Alternate Format

#### Basic Features

Name post-processing in the formatting hooks (Section 2.4.2) only affects the text. Continental formatting occurs in both the text and in the index. Therefore you need to use control sequences in the naming macro arguments.

Section 2.3.7 showed us that changing a control sequence will change a name, even if one cannot see the difference. Those changes must be consistent in the index to avoid spurious entries. Here is how we address that.

**3.1** We use `\AltFormatActive` at the start of this section to enable alternate formatting and switch it “on.” We begin with basic examples that do not change. We then move to advanced features that allow change in the text.

how to break stuff



If made the  $\langle SNN \rangle$  argument of a name macro, `\textsc{a Name, Problem}` will cause an error due to using commas as suffix delimiters. We fix that by using: `\textsc{a Name}, \textsc{Problem}`.

`\CapThis` still can break `\textsc{a Name}, \textsc{Problem}` under the normal formatting regime. Alternate formatting prevents this by suppressing the normal effects of `\CapThis`.

Previous methods to get Continental formatting still should work. Simply use the `altformat` option or `\AltFormatActive` to add protection against `\CapThis`.

`\AltFormatActive`

Both the `altformat` option and `\AltFormatActive` globally enable alternate formatting and switch the formatting macros “on.” This causes `\CapThis` only to work via `\AltCaps`.

`\AltFormatActive*`

When one wants to enable alternate formatting but keep the formatting macros in the “off” state, use the starred form `\AltFormatActive*`. It can change the effects of both the `altformat` option and `\AltFormatActive`. It causes `\CapThis` only to work via `\AltCaps`.

`\AltFormatInactive`

When one needs to switch alternate formatting “off” and deactivate its mechanism, use `\AltFormatInactive` to revert globally to standard formatting and the normal function of `\CapThis`.

|                                 | Enabled | Switched “On” |
|---------------------------------|---------|---------------|
| <code>\AltFormatActive</code>   | Yes     | Yes           |
| <code>\AltFormatActive*</code>  | Yes     | No            |
| <code>\AltFormatInactive</code> | No      | No            |

`\textSC`

Continental formatting can be as simple as using the short macro `\textSC`.

`\textIT`

Three other macros also implement alternate formatting. These macros make

`\textBF`

changes only when alternate formatting is active. We sort the index entry and

`\textUC`

demonstrate the formatting activated by `\AltFormatActive`.

```
\PretagName[Greta]{\textSC{Garbo}}{Garbo, Greta}
\PretagName[Ada]{\textIT{Lovelace}}{Lovelace, Ada}
\PretagName[Charles]{\textBF{Babbage}}{Babbage, Charles}
\PretagName{\textUC{Tokugawa}, Ieyasu}{Tokugawa Ieyasu}
```

```

\Name[Greta]{\textSC{Garbo}} ..... Greta GARBO; GARBO
\Name[Ada]{\textIT{Lovelace}} ..... Ada Lovelace; Lovelace
\Name[Charles]{\textBF{Babbage}} ..... Charles Babbage; Babbage
\Name{\textUC{Tokugawa}, Ieyasu}.. TOKUGAWA Ieyasu; TOKUGAWA

```



Formatting also occurs in the index using this method. Any time that a naming macro writes to the index, the flags that control these formatting macros must be in the same state, or else you will get spurious index entries.

comma karma

A comma delimiter splits the mandatory macro argument into a root and an affix. To avoid errors, format the name and suffix separately. The example below gives us **John David ROCKEFELLER III**, then **ROCKEFELLER**.

```

\PretagName[John David]{\textSC{Rockefeller},\textSC{III}}%
{Rockefeller, John David 3}
\begin{nameauth}
  \< JRIII & John David & \textSC{Rockefeller},\textSC{III} & >
\end{nameauth}

```

For non-Western names, the new syntax and the older syntax produce the same control sequence that identifies names. Again we are careful to avoid putting the comma delimiter within a container macro.

```

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

```

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

Only the new syntax allows alternate names to be used in the text. Thus, `\LFukuyama[Sensei]` **FUKUYAMA Sensei** wrote *Nihon Fukuin Rūteru Kyōkai Shi* in 1954, after studying in the US in the 1930s. The old syntax `\LOFukuyama[Sensei]` yields **FUKUYAMA Takeshi**[Sensei].

## Advanced Features

A more complex version of alternate formatting allows us to make format changes in the text while keeping format consistent in the index. We use `\textSC`, `\textIT`, `\textBF`, and `\textUC` with `\noexpand` and special triggering macros. Using `\noexpand` is crucial because we do not want to have the macros expand at the wrong time, giving us the wrong results. Thus:

|  |                 |
|--|-----------------|
| <code>\Name[Martin]{\textSC{Luther}}</code>          | <i>basic</i>    |
| <code>\Name[Martin]{\noexpand\textSC{Luther}}</code> | <i>advanced</i> |

Remember `\textsc{a Name}`, `\textsc{Problem}`? With a little work adding the alternate formatting macros and `\noexpand` we get:

```

\noexpand\textSC{\noexpand\AltCaps{a} Name}, \noexpand\textSC{Problem}

```

With an additional change to the formatting hooks, whenever alternate formatting is active, the naming macros will avoid **A NAME PROBLEM**. **A Name Problem** will not occur even with `\CapThis` and **a Name** will work just fine. We suppressed the index entries that would have been created here.

The macros below work together for advanced alternate formatting.

- |                      |  |
|----------------------|--|
| <code>\AltOff</code> | 1. The macro <code>\AltOff</code> does nothing except when called in a formatting hook, where it “switches off” alternate formatting. When that happens, <code>\textSC</code> , <code>\textBF</code> , <code>\textIT</code> , and <code>\textUC</code> do nothing. This macro works with the <code>altformat</code> option and when <code>\AltFormatActive</code> has been called. |
| <code>\AltOn</code>  | 2. The macro <code>\AltOn</code> does nothing except when called in a formatting hook, where it “switches on” alternate formatting. When that happens, <code>\textSC</code> , <code>\textBF</code> , <code>\textIT</code> , and <code>\textUC</code> perform their changes. This macro works when <code>\AltFormatActive*</code> has been called.                                  |
|                      | 3. Using <code>\noexpand</code> is the golden key ( <i>clavis aurea</i> ) that lets us expand formatting changes only when desired. It enables this kind of formatting hook, which we must implement:  |

```
\renewcommand*\MainNameHook{\AltOff}
```

- |                       |  |
|-----------------------|--|
| <code>\AltCaps</code> | 4. Since the normal effects of <code>\CapThis</code> are disabled <code>\AltCaps</code> provides an alternate means to this end. It capitalizes its argument in braces <code>{ }</code> when it is used in a macro hook and triggered by <code>\CapThis</code> . |
|-----------------------|--|

Since we used `\AltFormatActive` in this section it has triggered formatting by default. We only need to change `\MainNameHook` and `\FrontNameHook` because we want to have formatting in first uses but suppress it in subsequent uses. Below we match the style of this manual with the redesign of the formatting hooks and we include a sample text:

```
\renewcommand*\MainNameHook[1]%
{\color{naorange}\sffamily\AltOff}
```

With the 500th anniversary of the Reformation in 2017, studies should focus both on the life of **Martin LUTHER** and on the social, religious, and political factors of the time that influenced **Luther**.

We show alternate formatting and capitalization in the text, here being mindful of how medieval Italian differs from modern Italian:

```
\begin{nameauth}
< Cath & Catherine \noexpand\AltCaps{d}e'
& \noexpand\textSC{Medici} & >
\end{nameauth}
```

This gives us **Catherine de' MEDICI** and **Medici**. To get either **De' MEDICI** or **De' Medici**, use `\CapThis\LCath[\noexpand\AltCaps{d}e']`.

Sections 2.4.2 and 2.9.7 have more on these topics, especially the use of `\NameParser`. We return to normal formatting with `\AltFormatInactive` and ensure that no names in the alternate regime are used elsewhere.

Back to Section 1.6

## 2.5 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. Some aspects of indexing go beyond the scope of this package.<sup>17</sup>

### 2.5.1 Indexing Control

`\IndexActive` 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. `\ExcludeName` and `\IncludeName` do not deactivate indexing, but they leverage the cross-referencing system to prevent page entries.

`\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.

**`\IndexInactive` suppresses index sorting and tagging macros.**

`\SkipIndex` The prefix macro `\SkipIndex` will suppress indexing for just one instance of a naming or cross-referencing macro. It will not alter name forms or formatting. For example, `\SkipIndex\Name[Monty]{Python}` produces `Monty Python` in the text with no index entry. The same thing again yields `Python`. Both `\IndexName` and `\IndexRef` ignore `\SkipIndex` and allow its effect, with other prefix macros, to “pass through” to the next naming macro.

`\JustIndex` This prefix macro makes `\Name` and `\Fname` act just like a call to `\IndexName` one time only. That means, like `\IndexName`, the effects of all the other prefix macros will “pass through” to the next naming macro. Both `\AKA` and `\PName` ignore and reset the flag controlled by this macro.

All the changes made by the prefix macros pass through `\JustIndex\⟨name1⟩` to the next instance of `\Name`, etc., `\⟨name2⟩`. This is exactly as if you called `\IndexName`. This makes `\JustIndex\⟨name1⟩\SkipIndex\⟨name2⟩` equivalent to `\SkipIndex\JustIndex\⟨name1⟩\⟨name2⟩`.

Now we use tricks from Sections 2.5.2, 2.5.3 and 2.7.2 to modify name forms, formatting, and indexing. Instead of using `\SkipIndex`, `\IndexInactive`, and `\IndexActive`, here we let the name exclusion mechanism protect a name:

```
\begin{nameauth}
  < Washs & George & Washington's & >
\end{nameauth}
\ExcludeName[George]{Washington's}
```

`\Washs` and `\Washs` produce `George Washington's` and `Washington's`, but no index entries. Use `\JustIndex\Wash` as needed.

Back to Section 1.6

---

<sup>17</sup>The maintainer of the `babel` package is working to address issues with Roman page numbers. For memoir, search for “memoir babel index” at <http://tex.stackexchange.com>.

## 2.5.2 Index Entries

`\IndexName` The naming macros (`\Name`, etc.) use this macro to create index entries. You can use it too. 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. If `⟨FNN⟩` are absent, `\IndexName` sees `⟨Alternate names⟩` as an affix or Eastern forename using the older 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 [Mulvany, 152–82] and the *Chicago Manual of Style* regarding Western name affixes. Thus **Chesley B. Sullenberger III** becomes “Sullenberger, Chesley B., III” in the index.

To show what gets into the index entries, consider the following example, much of which gets set up only once in the document.

```
\begin{nameauth}
  \< Dem      &          & Demetrius, I      & >
  \< Harnack & Adolf    & Harnack          & >
  \< JWG      & J.W. von & Goethe          & >
  \< Miyaz    &          & Miyazaki, Hayao & >
\end{nameauth}
```

We add a text tag as a sobriquet and use the hook from Section 2.3.6:

```
\NameAddInfo{Demetrius, I}{Soter}
\makeatletter\renewcommand*\NamesFormat[1]{\begingroup%
\protected@edef\temp{\endgroup{\color{naviolet}\sffamily #1 %
\noexpand\NameQueryInfo[\unexpanded\expandafter{\the\@nameauth@toksa}}
{\unexpanded\expandafter{\the\@nameauth@toksb}}}
[\unexpanded\expandafter{\the\@nameauth@toksc}]}}\temp\makeatother
```

We also add an index tag: `\TagName{Demetrius, I}{ Soter, king}` and a sort tag: `\PretagName{Demetrius, I}{Demetrius 1}`.

| Text                  | Source                             | Index                   |
|-----------------------|------------------------------------|-------------------------|
| Demetrius I Soter     | <code>\LDem</code>                 | Demetrius I Soter, king |
| Demetrius I           | <code>\LDem</code>                 | Demetrius I Soter, king |
| Adolf von Harnack     | <code>\LHarnack[Adolf von]</code>  | Harnack, Adolf          |
| Adolf Harnack         | <code>\LHarnack</code>             | Harnack, Adolf          |
| Joh. Wolfg. v. Goethe | <code>\LJWG[Joh. Wolfg. v.]</code> | Goethe, J.W. von        |
| J.W. von Goethe       | <code>\LJWG</code>                 | Goethe, J.W. von        |
| Miyazaki Hayao        | <code>\LMiyaz</code>               | Miyazaki Hayao          |
| Miyazaki Sensei       | <code>\LMiyaz[Sensei]</code>       | Miyazaki Hayao          |

Everything in the `⟨FNN⟩` and `⟨SNN⟩` arguments, including the `⟨Affix⟩`, gets in the index. When the final optional argument is interpreted as an alternate name, it does not become part of the index entry. Text tags never get in the index, but index tags always get in the index.

### 2.5.3 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. Section 2.7.2 show how cross-references are independent of other data sets. The syntax is:

`\IndexRef[\langle FNN \rangle]{\langle SNN \rangle}[\langle Alternate names \rangle]{\langle reference target \rangle}`

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}`. To format that reference in the text, use `\AKA` (Section 2.8).

Please see page 44 regarding complex cross-references.

**\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.<sup>18</sup> However, this should be used with caution, as the following points indicate:

- If on page 10 there is `\SeeAlso\IndexRef{Bar}{Foo}`, one *cannot* have index page entries for “Bar” thereafter. A *see also* reference comes after page references.
- If on page 10 there is `\SeeAlso\IndexRef{Bar}{Foo}`, one *can* have index page entries for “Foo” thereafter because it is the target of “Bar.”
- If on page 10 there is `\Name{Bar}` and on page 12 `\IndexRef{Bar}{Foo}`, that will not work because *see* references cannot contain page references.
- Suggestion: Group references together: `\IndexRef{Bar}{Baz; Foo}`. Avoid `\IndexRef{Bar}{Baz} \IndexRef{Bar}{Foo}`.<sup>19</sup>

`\IndexRef` causes an index tag with the format `\langle some text \rangle | \langle some macro \rangle` to be reduced to `\langle some text \rangle` in the cross-reference. This allows cross-references to work with any index macro, e.g. `|hyperpage`, used by `\TagName` (Section 2.5.5).

**\ExcludeName** 3.0 This macro prevents a name from being used as either an index entry or as an index cross-reference. It ignores extant cross-references. The syntax is:

`\ExcludeName[\langle FNN \rangle]{\langle SNN \rangle}[\langle Alternate names \rangle]`

After `\ExcludeName[Kris]{Kringle}`, you can use `\Name[Kris]{Kringle}` to get **Kris Kringle** and **Kringle**. After `\ExcludeName[Santa]{Claus}` you can use `\AKA[Kris]{Kringle}[Santa]{Claus} Santa Claus`. No index entries are created.

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.

<sup>18</sup>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.

<sup>19</sup>Professional indexers often use programs like `Cindex` that enforce a rigorous, standard methodology and syntax. The `nameauth` package likewise tries to follow suit.



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

**3.0**

```
\IncludeName [\langle FNN \rangle]{\langle SNN \rangle}[\langle Alternate names \rangle]
\IncludeName*[\langle FNN \rangle]{\langle SNN \rangle}[\langle Alternate names \rangle]
```

The unstarred form of `\IncludeName` only removes an exclusion created by `\ExcludeName`. The starred form of `\IncludeName` completely unprotects a cross-reference and allows it to have a page entry like a name.

For example, we used `\ExcludeName{Attila, the Hun}` after his appearance in Section 1.4. Using `\IfAKA{Attila, the Hun}` (Section 2.7.1) tells us that, “Attila is excluded.” Now if we `\IncludeName{Attila, the Hun}`, a reference to `\LAttil` will create a name and an index entry on this page: **Attila the Hun**. `\IfAKA` now tells us that “Attila is a name.”

Cross-references get more protection. `\IfAKA[Jay]{Rockefeller}` (a reference in a footnote from Section 1.4) tells us that “Jay is a cross-reference.” Using `\IncludeName[Jay]{Rockefeller}` changes nothing: we still get “Jay is a cross-reference.” `\IncludeName*[Jay]{Rockefeller}` results in “Jay is a name,” removing all protection of that cross-reference.

Back to Section 1.6

## 2.5.4 Index Sorting

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

### Basic Sorting (for Makeindex and More)

`\PretagName`      The `nameauth` package integrates this sort of index sorting automatically by using a “pretag.” Section 2.7.2 show how sorting tags are independent of other data sets in `nameauth`. The syntax is:

**2.0**

```
\PretagName [\langle FNN \rangle]{\langle SNN \rangle}[\langle Alternate names \rangle]{\langle tag \rangle}
```

`\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}{Łukasiewicz, 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 tagged and sorted automatically.

Additionally, one can include sub-entry delimiters when sorting, so `\langle Some Name \rangle` can be sorted as a sub-entry of “MyCategory” by the following:

```
\PretagName[Some]{Name}{MyCategory!Name, Some}
```

One also can “pretag” a cross-reference created with `\IndexRef`, `\AKA`, and so on. See also Sections 2.5.3 and 2.8.



Although the `\PretagName` macro might look similar to the other tagging macros, its use is quite 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 added space) and `Æ` becomes `\IeC_{\AE_}` (two added spaces).

Section 2.9.3 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. Thus we have the following:

```
NFSS: \index{Fußball} → \indexentry{Fu\IeC_{\ss_}ball}{\langle page\rangle}
fontspec: \index{Fußball} → \indexentry{Fußball}{\langle page\rangle}
cseq: \index{Fu\ss ball} → \indexentry{Fu\ss_ ball}{\langle page\rangle}
```

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 and modifying manual index entries with `nameauth`:

```
\newcommand\IndexExample[1]{%
  \protected@edef\argument{#1}\index{\argument}}%
\IndexExample{\textsc{football}} →
  \indexentry{\textsc_ _{football}}{\langle page\rangle}
\index{\textsc{football}} →
  \indexentry{\textsc{football}}{\langle page\rangle}
```

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
```

Back to Section 1.6

### 2.5.5 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}
```

Index tags are not “pretags.” Section 2.7.2 show how index tags are independent of other data sets in `nameauth`. To help sort that out, we look at what parts of the argument of `\index` get affected by these commands:

|                                  |                          |                          |                      |                                      |
|----------------------------------|--------------------------|--------------------------|----------------------|--------------------------------------|
| <code>\index{Aethelred 20</code> | <code>\PretagName</code> | <code>Ethelred II</code> | <code>, king}</code> | <code>\TagName and \UntagName</code> |
|----------------------------------|--------------------------|--------------------------|----------------------|--------------------------------------|

All the tagging commands are keyed to the name arguments. `\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. `\TagName` causes the `nameauth` indexing macros to append “`,pope”` to the index entries for the popes below:

```
\TagName{Leo, I}{, pope}
\TagName{Gregory, I}{, pope}

\Name{Leo, I} was known as \AKA{Leo, I}{Leo, the Great}.
\Name{Gregory, I} was known as \Name{Gregory, I}
‘‘\ForceFN\AKA*{Gregory, I}{Gregory}[the Great].’’

Leo I was known as Leo the Great.
Gregory I was known as Gregory “the Great.”
```

We see both the old syntax and the new syntax used above. `\TagName` works with all name types, but not with cross-references from `\IndexRef`, etc. Tags are literal text that can be daggers, asterisks, and so on. 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.



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 with `ltxdoc` and `hypdoc`. You may have to use `\string|hyperpage` where a vertical bar is active, as in `ltxdoc`.

For example, `\newcommand\orphan[2]{#1}` allows one to use `|orphan{\text}` in an index tag to replace the page number with `\text`. The `idx` file will contain `\indexentry{\name}|orphan{\text}}{\page}`. The `ind` file will have something like `\item \name, \orphan{\text}}{\page}`, depending on the index style.

`\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 2.7.2:

```
This refers to \Name[John]{Smith}.\
Now we have a new \TagName[John]{Smith}{ (second)}%
\ForgetThis\Name[John]{Smith}.\
Now we have a third \TagName[John]{Smith}{ (third)}%
\ForgetThis\Name[John]{Smith}.\
Then back to the first \UntagName[John]{Smith}\Name*[John]{Smith}.

This refers to John Smith.
Now we have a new John Smith.
Now we have a third John Smith.
Then back to the first John Smith.
```

The tweaking macros (Section 2.7.2) make it seem like you are dealing with three people who have the same name. The index tags will group together those entries that have the same tag.<sup>20</sup>

Back to Section 1.6

## 2.6 “Text Tags”

Section 2.5.5 deals with similar tagging features in the index. “Text tags” are not printed automatically with every name managed by `nameauth`. Section 2.7.2 show how text tags are independent of other data sets. Section 2.9.6 offers additional examples on using these macros.

Several major uses include optional sobriquets, life dates, regnal dates, footnotes, biographical vignettes, margin paragraphs, and so on.

`\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.

---

<sup>20</sup>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 re-tags him with an asterisk again. We also used `\string\hyperpage` in all the index tags. The information is not shown above for the sake of simplicity and pedagogy.

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

```
\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. In text tags one might use asterisks, daggers, and even footnotes, such as one for [Schuyler Colfax](#).<sup>21</sup> We can include a “text tag” within another one, thus building complex relations. Keeping this in mind, we look at the source for the footnote:

```
\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}
```

`\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⟩]
```

`\NameClearInfo[George]{Washington}` will cause the next attempt at making a query, `\NameQueryInfo[George]{Washington}`, to produce nothing.

Back to Section [1.6](#)

## 2.7 Name Decisions

### 2.7.1 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*`.

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 [2.9.6](#) and [2.9.7](#) for related ideas about tokens and expansion. Using `\tracingmacros`, `\show`, or `\meaning` can help you.

---

<sup>21</sup>Seventeenth vice-president of the US during the first term (1869–73) of [Ulysses S. Grant](#) (president 1869–77).

`\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” because we have yet to invoke the name `\Name[Bob]{Hope}`. We will create a manual index entry here.

```
\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 [Elizabeth I](#), we get “I met Bess” with a similar example:

```
\IfMainName{Elizabeth, I}{I met Bess}%
{I have not met Bess}
```

`\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 2.4.2](#), 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 2.7.2](#) and [2.9.2](#) to understand the scope and operation of main- and front-matter names.

This space intentionally left blank.

`\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 2.5.3).

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 2.5.3) 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}
```

By using the text tag macros with the conditional macros, one can display information associated with a name based on whether or the name has occurred. Below we disable indexing with `\IndexInactive`:

```
\NameAddInfo{Sam}
{\IfMainName{Freddy}%
  {\SkipIndex\Name{Freddy's} sidekick}%
  {a young gardener helping his granddad}}

There is \Name{Sam}. He is \NameQueryInfo{Sam}.
Then \Name{Sam} met \Name{Freddy}, who lives with his uncle.
Now he is \NameQueryInfo{Sam} on a quest to save the realm.

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

We use `\SkipIndex` to prevent the name “Freddy's” from making an index entry of its own. See Section 2.5.1. Take care to avoid a stack overflow by conditionally nesting tags “down the rabbit hole.”

Back to Section 1.6

## 2.7.2 Changing Decisions

The following summary of macros that can change (not just read) different data sets will help us put this section into better perspective:

| Macro                                    | Names | Xrefs | Sort<br>Tag | Index<br>Tag | Text<br>Tag |
|--|-------|-------|-------------|--------------|-------------|
| <code>\Name \Name* \FName</code>         | Yes   | No    | No          | No           | No          |
| <code>\ForgetName \SubvertName</code>    | Yes   | No    | No          | No           | No          |
| <code>\PName \PName*</code>              | Yes   | Yes   | No          | No           | No          |
| <code>\AKA \AKA* \IndexRef</code>        | No    | Yes   | No          | No           | No          |
| <code>\ExcludeName</code>                | No    | Yes   | No          | No           | No          |
| <code>\IncludeName \IncludeName*</code>  | No    | Yes   | No          | No           | No          |
| <code>\PretagName</code>                 | No    | No    | Yes         | No           | No          |
| <code>\TagName \UntagName</code>         | No    | No    | No          | Yes          | No          |
| <code>\NameAddInfo \NameClearInfo</code> | No    | No    | No          | No           | Yes         |

The macros in this section force either a first or subsequent use, helpful especially with overlays in the `beamer` class. They do not affect `\AKA` and `\PName`. They always are global with respect to  $\text{\LaTeX}$  scoping rules.

“Forgetting” a name not only changes its format, but also its displayed form and its status with decision macros. Sometimes you want all the changes (`beamer` overlays) and sometimes not (use `\Name*`, `\ForceName`, etc.).

|                       | Name Length   | Format Hooks | Decision <sup>22</sup> |
|-----------------------|---------------|--------------|------------------------|
| <b>First Use</b>      | Always long   | First        | False                  |
| <b>Subsequent Use</b> | Long or short | Subsequent   | True                   |

`\ForgetName` This macro takes the same arguments as `\Name`. It ignores alternate names if  $\langle FNN \rangle$  are present. It “forgets” a name, forcing a “first use” The syntax is:

`\ForgetName[\langle FNN \rangle]{\langle SNN \rangle}[\langle Alternate names \rangle]`

`\ForgetThis` This mode switch causes the next instance of a naming macro or shorthand to call `\ForgetName` internally. After knowing `\Einstein` “**Einstein**” we forget him and again have a first reference: `\ForgetThis\Einstein` “**Albert Einstein**.”

`\SubvertName` This macro is the opposing analogue of the macros that we saw above. It “subverts” a name, forcing a “subsequent use.” The syntax is:

`\SubvertName[\langle FNN \rangle]{\langle SNN \rangle}[\langle Alternate names \rangle]`

`\SubvertThis` This mode switch causes the next instance of a naming macro or shorthand to call `\SubvertName` internally. `\ForgetThis` takes precedence over `\SubvertThis`.

|                                 |                         |  |
|---------------------------------|-------------------------|--|
| <code>\SubvertThis\LANth</code> | <b>Susan B. Anthony</b> | <i>forced subsequent use, forced long</i>  |
| <code>\ForceName\SAnth</code>   | <b>Susan B.</b>         | <i>subsequent use, forced first format</i> |
| <code>\ForgetThis\SAnth</code>  | <b>Susan B. Anthony</b> | <i>forced first use and format</i>         |
| <code>\SAnth</code>             | <b>Susan B.</b>         | <i>subsequent use, short</i>               |

We met `\ForceName` back in Section 2.4.2. Here we use it with a subsequent name use to format it as a first use. We will meet `\ForceName` again in Section 2.8.

Naming system scope By default, these macros affect a name form in both front matter and main matter naming systems. The example on page 37 above gave us the answer, “**Carnap** is both”. After we use `\ForgetName[Rudolph]{Carnap}` we get the result: “**Rudolph Carnap** is not mentioned.” Both front- and main-matter names were forgotten and now we have a first-use situation.

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 2.4.2), 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. This manual uses that behavior to synchronize uses of names between formatting systems.

`\LocalNames` If this default behavior is not desired, `\LocalNames` restricts the macros above to the current naming system. After `\LocalNames`, if you are in a “front matter” section (the `frontmatter` option or `\NamesInactive`) the macros above will affect only names in that section. The same is true if you are in a “main matter” section via the `mainmatter` option or `\NamesActive`. `\GlobalNames` restores the default behavior. Remember that this is respective to formatting systems, not document scope! Section 2.9.2 goes into greater detail on system-level scoping.

Back to Section 1.6

<sup>22</sup>Decision outcome prior to the name being used.



## 2.8 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 the macros here completely. Yet here they are, if needed.

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

```
\AKA [ $\langle FNN \rangle$ ]{ $\langle SNN \rangle$ }[ $\langle Alt. FNN \rangle$ ]{ $\langle Alt. SNN \rangle$ }[ $\langle Alt. names \rangle$ ]
\AKA* [ $\langle FNN \rangle$ ]{ $\langle SNN \rangle$ }[ $\langle Alt. FNN \rangle$ ]{ $\langle Alt. SNN \rangle$ }[ $\langle Alt. names \rangle$ ]
```

Both macros create a cross-reference in the index from the  $\langle Alt. FNN \rangle$ ,  $\langle Alt. SNN \rangle$ , and  $\langle Alt. names \rangle$  fields to a target defined by  $\langle FNN \rangle$  and  $\langle SNN \rangle$ , regardless of whether that name exists. **The name order for `\AKA` is opposite that of `\IndexRef`.**<sup>23</sup> See also Section 2.5.5.

`\AKA` only prints whatever form of name in the text that you manually specify. It is designed for the occasional mention of alternate names. See page 44 for alternate solutions. `\SeeAlso` works with `\AKA`, `\AKA*`, and `\PName`.

`\AKA` prints the  $\langle Alt. FNN \rangle$  and  $\langle Alt. SNN \rangle$  fields in the body text. If the  $\langle Alt. names \rangle$  field is present, `\AKA` swaps it with the  $\langle Alt. FNN \rangle$  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`.

### General Tips

- [ $\langle FNN \rangle$ ]{ $\langle SNN \rangle$ } is the target. [ $\langle Alt. FNN \rangle$ ]{ $\langle Alt. SNN \rangle$ }[ $\langle Alt. names \rangle$ ] is the cross-reference to the target. Neither create page references.
- The older non-Western syntax cannot be used with [ $\langle FNN \rangle$ ]{ $\langle SNN \rangle$ }. It can be used with { $\langle Alt. SNN \rangle$ }[ $\langle Alt. names \rangle$ ], but we discourage that.
- Only the  $\langle SNN \rangle$  and  $\langle Alt. SNN \rangle$  fields use comma-delimited suffixes.
- One cannot create an index tag for a cross-reference, but one can sort that reference with `\PretagName`.
- [ $\langle Alt. FNN \rangle$ ]{ $\langle Alt. SNN \rangle$ }[ $\langle Alt. names \rangle$ ] in `\AKA` correspond to the name fields in `\PretagName`.
- **Jimmy Carter** is not a cross-reference when it takes a form like:  
`\DropAffix\Name*[J.E.]{Carter, Jr.}[Jimmy]`.
- **Jimmy Carter** is a cross-reference when it takes a form like:  
`\AKA[J.E.]{Carter, Jr.}[Jimmy]{Carter}`.
- To index stage names:  
`\Name[The Amazing]{Kreskin}..... The Amazing Kreskin`  
`\AKA[The Amazing]{Kreskin} [Joseph]{Kresge} ... Joseph Kresge`
- To keep stage names out of the index (index entries suppressed):  
`\Name[J.]{Kreskin}[The Amazing] ..... The Amazing Kreskin`  
`\AKA[J.]{Kreskin}[Joseph]{Kresge} ..... Joseph Kresge`

<sup>23</sup>That ordering is due to the collision between  $\langle Alt_1 \rangle$  and  $\langle FNN_2 \rangle$  in a hypothetical `\AKA[ $\langle FNN_1 \rangle$ ]{ $\langle SNN_1 \rangle$ }[ $\langle Alt_1 \rangle$ ][ $\langle FNN_2 \rangle$ ]{ $\langle SNN_2 \rangle$ }[ $\langle Alt_2 \rangle$ ]` By only allowing  $\langle FNN_1 \rangle$  and  $\langle SNN_1 \rangle$  for the target name, we can let the other fields permit an unrestricted cross-reference.

## Examples


We make cross-references to **Bob Hope**, where sll of the forms below create the cross-reference “Hope, Leslie Townes *see* Hope, Bob”:

|  |                     |
|--|---------------------|
| <code>\AKA[Bob]{Hope}[Leslie Townes]{Hope}</code>                      | Leslie Townes Hope  |
| <code>\RevComma\AKA[Bob]{Hope}[Leslie Townes]{Hope}</code>             | Hope, Leslie Townes |
| <code>\AKA[Bob]{Hope}[Leslie Townes]{Hope}%<br/>[\ignorespaces]</code> | Hope                |
| <code>\AKA[Bob]{Hope}[Leslie Townes]{Hope}[Lester T.]</code>           | Lester T. Hope      |
| <code>\AKA*[Bob]{Hope}[Leslie Townes]{Hope}</code>                     | Leslie Townes       |
| <code>\AKA*[Bob]{Hope}[Leslie Townes]{Hope}[Lester]</code>             | Lester              |

Next we see what happens with references to **Louis XIV**, **Lao-tzu**, and **Gregory I**, as well as **Lafcadio Hearn** and **Charles du Fresne**:

|   |                   |
|---|-------------------|
| <code>\AKA{Louis, XIV}{Sun King}</code>                     | Sun King          |
| <code>\AKA*{Louis, XIV}{Sun King}</code>                    | Sun King          |
| <code>\AKA{Lao-tzu}{Li, Er}</code>                          | Li Er             |
| <code>\AKA*{Lao-tzu}{Li, Er}</code>                         | Li                |
| <code>\AKA[Charles]{du Fresne}{du Cange}</code>             | du Cange          |
| <code>\CapThis\AKA[Charles]{du Fresne}{du Cange}</code>     | Du Cange          |
| <code>\CapName\AKA[Lafcadio]{Hearn}{Koizumi, Yakumo}</code> | KOIZUMI Yakumo    |
| <code>\RevName\AKA[Lafcadio]{Hearn}{Koizumi, Yakumo}</code> | Yakumo Koizumi    |
| <code>\AKA{Gregory, I}{Gregory}[the Great]</code>           | Gregory the Great |
| <code>\AKA*{Gregory, I}{Gregory}[the Great]</code>          | Gregory           |
| <code>\ForceFN\AKA*{Gregory, I}{Gregory}[the Great]</code>  | the Great         |

## Formatting Alternate Names: General

**formatAKA**  `\AKA` and its derivatives use the subsequent-use formatting hooks `\MainNamesHook` and `\FrontNamesHook`. This was designed originally to keep cross-references from looking like main names by accident when they were intriduced in the body text. In order to be freed of those constraints, use the **formatAKA** package option. Note the caveats that come therewith.

We show **formatAKA** used with `\AKA{Elizabeth, I}[Good Queen]{Bess}`. The colors indicate which formatting hooks are being used.

**Front Matter:** Elizabeth I was known as “Good Queen Bess.”  
Again we mention Queen Elizabeth, “Good Queen Bess.”  
`\ForceName:` Good Queen Bess.

**Main Matter:** Elizabeth I was known as “Good Queen Bess.”  
Again we mention Queen Elizabeth, “Good Queen Bess.”  
`\ForceName:` Good Queen Bess.

Section 2.7.2 also shows how cross-references are independent of other data sets in **nameauth**. Cross-references do not respect the two namng systems. The first time that the cross-reference appears, we see that **formatAKA** lets it use the first-use hooks. Thereafter, it uses the subsequent-use hooks. When we switched to the main matter, the cross-reference **Good Queen Bess** did not switch to a first use until we used `\ForceName`. Now we compare the **alwaysformat** option:

**Front Matter:** Elizabeth I was known as “Good Queen Bess.”  
 Again we mention Queen Elizabeth, “Good Queen Bess.”  
`\ForceName: Good Queen Bess.`

**Main Matter:** Elizabeth I was known as “Good Queen Bess.”  
 Again we mention Queen Elizabeth, “Good Queen Bess.”  
`\ForceName: Good Queen Bess.`

With `alwaysformat`, all the names in the document use only the first-use hooks, never the subsequent-use hooks. This option tends to get little use in the newer versions of `nameauth`. It was more useful in early versions when `\NamesFormat` was the only formatting hook.

### Formatting Alternate Names: Continental

The following annotated example shows the simple Continental form that we introduced in Section 2.4.3. We initiate the alternate formatting framework with `\AltFormatActive` and take care not to use the names below outside of it.

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, but we do not start with that. We begin with `\AKA*` in order to use his legal name as an alias for his more popular stage name. `\AKA*` prints “Heinrich Wilhelm” in the body text and sets up the index cross-reference “RÜHMANN, Heinrich Wilhelm *see* RÜHMANN, Heinz.”  
`\AKA*[Heinz]{\textSC{Rühmann}}%`  
`[Heinrich Wilhelm]{\textSC{Rühmann}} %`
3. `\SubvertThis` makes `\FName` print “Heinz.”  
`\SubvertThis‘‘\FName[Heinz]{\textSC{Rühmann}}’’ %`
4. `\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.

Of course, this example is but one among a number of solutions. The point is to find a solution that best fits one’s needs. We now resume normal formatting with `\AltFormatInactive`.

## Advanced Cross-Referencing

- 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:<sup>24</sup>

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

`\AKA` and `\AKA*` never create never page entries. When the alternate name needs to be indexed with page numbers and *see also* references, do the following:

- Refer to the person intended, *e.g.*:  
`Maimonides` (`Moses ben-Maimon`):  
`\Name{Maimonides} (\AKA{Maimonides}{Moses ben-Maimon})`
- We now have a main name with a page entry, as well as a “*see* reference” name. If we fail to refer to the main name, we would have a cross-reference to an entry that does not exist.
- Before creating a *see also* cross-reference, one must refer to the alternate name so that all the page entries precede the cross-reference, *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:<sup>25</sup>  
`\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 2.5.4) helps to avoid the need for manual index entries, as the following example shows:

```
\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, use it as such. Section 2.3.6 talks about how one can use `\ForceFN\FName{Thomas, Aquinas}` to refer to Aquinas. Using `\ForceFN\Name{Thomas, Aquinas}` will produce Thomas. That helps prevent unwanted side effects with Eastern names.

---

<sup>24</sup>We shorten the index entries via `\Name[W.]{Snel van Royen}[Willebrord]`, and for his son, `\Name[R.]{Snel van Royen}[Rudolph]`.

<sup>25</sup>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>.

`\PName` These macros were meant for Western names and developed in the early versions of `nameauth`. They no longer fit well with the package. They print a main name followed by a cross-reference in parentheses, the syntax being:

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

Apart from `\SkipIndex`, prefix macros only work on the name given by `⟨FNN⟩` and `⟨SNN⟩`, not on the latter cross-reference. `\SkipIndex` keeps both names out of the index. Below we see the only name types that this macro 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}}{\textit{Doctor mellifluus}}
\PName{Bernard, of Clairvaux}{\textit{Doctor mellifluus}}
                                           Bernard of Clairvaux (Doctor mellifluus)
                                           Bernard (Doctor mellifluus)
```

Like `\AKA`, `\PName` cannot use the older 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**) and **William** (**William the Conqueror**).<sup>26</sup> If you use `\PName*`, again you will get the long reference **William I** (**William the Conqueror**).

`\PName*{William, I}[William]{the Conqueror}` puts “**William I** (**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. We suppressed the index entry with `\SkipIndex`.

Back to Section 1.6

## 2.9 Longer Examples

### 2.9.1 Variant Names

**3.1** This section demonstrates how `nameauth` helps one manage a name authority. Handling name variants has become easier than before. We start with some simple cases and move on to complex ones:

- Where Iron Mike occurs in the text, include `\IndexName[Mike]{Tyson}`.
- `\SubvertThis\FName[Mike]{Tyson}[Iron Mike]` always prints **Iron Mike** indexed as “Tyson, Mike”. That form uses the subsequent-use formatting hooks. `\ForceName\SubvertThis\FName[Mike]{Tyson}[Iron Mike]` prints **Iron Mike** with the first-use hooks.
- The form `\Iron Iron Mike Tyson` can be set up with:

```
\newcommand*\Iron{\SubvertThis\Name*[Mike]{Tyson}[Iron Mike]}
```

In `nameauth` it makes little sense to “force” the subsequent use because it is the common use. First uses are rare. That is why we set up the subsequent use with `\SubvertThis` and create a first use when needed with `\ForceName`. `\ForceName\Iron` prints **Iron Mike Tyson**, again indexed as “Tyson, Mike”.

<sup>26</sup>The form `\PName{William, I}{William}[the Conqueror]` works, but we discourage it. Also choose forms like `\PName{Lao-tzu}{Li, Er}` instead of `\PName{Lao-tzu}{Li}{Er}`. Avoiding the older syntax with `\AKA` and `\PName` avoids error.

- Use `\IndexRef{Iron Mike}{Tyson, Mike}` to create a *see* cross-reference from “Iron Mike” to “Tyson, Mike” in the index. Be sure to have an occurrence of `\Name[Mike]{Tyson}` in the text.
- 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.

When you want alternate names that can change form and format independently, do the following:

1. We start by deciding that the canonical name form we wish to use is “W.E.B. Du Bois.” We want to manage the alternate form “W.E.B. DuBois” as if it were an occurrence of the canonical name. We set up the name authority:

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

2. This name gives us an extra level of difficulty because the two variants differ only in terms of spaces. They share the same internal representation in the `nameauth` macros: `W.E.B. !DuBois`. We fix this ambiguity by inserting a non-printing control sequence in the alternate form, such as `{Du\empty Bois}`. That prevents “DuBois” from breaking at the end of a line or page. A discretionary hyphen would allow the name to break.<sup>27</sup>
3. Instead of using `\SkipIndex\AltDuBois` every time we wanted to avoid making an index entry, we create a cross-reference in the index from the alternate name to the canonical name:

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

From this point onward, no page entry for `W.E.B. DuBois` will occur in the index unless manipulated by `\IncludeName*`. The canonical `W.E.B. Du Bois` functions as a different name and is not affected.

**3.0** Indexing both name forms would be trivial. One can use both forms 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`.

**3.1** Indexing with the canonical name form `Du Bois` whenever we see `DuBois` is slightly more complicated:

- We no longer wrap each name automatically with two index entries, so we would need to keep track of page breaks and this alternate name.
- We could use `\JustIndex\DuBois\AltDuBois` to get `DuBois`.
- We could create macros based on that:
 

```
\global\newcommand*\OtherDuBois{\JustIndex\DuBois\AltDuBois}
\global\newcommand*\LOtherDuBois{\JustIndex\DuBois\LOAltDuBois}
\global\newcommand*\SOtherDuBois{\JustIndex\DuBois\SAltDuBois}
```

With `\ForgetThis\OtherDuBois` we get `W.E.B. DuBois` and `DuBois` thereafter. `\LOtherDuBois` gives us `W.E.B. DuBois`, while with `\SOtherDuBois` we get `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 work.

Back to Section 1.6

---

<sup>27</sup>Ignoring spaces in names is good because it aids fault tolerance, thereby decreasing spurious index entries. Here we have a special case where this behavior is not useful.

### 2.9.2 \LocalNames

As mentioned previously in Section 2.7.2, 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. To ignore any local scoping we use `\global\NamesInactive`:

```
\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
```



### 2.9.3 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 |
| Ò Ó Ô Õ Ö Ø     | Ù Ú Û Ü Ý       | Þ ß         |
| à á â ã ä å æ   | ç è é ê ë       | ì í î ï ð ñ |
| ò ó ô õ ö ø     | ù ú û ü ý       | þ ÿ         |
| Ă ă Ą ą Ć ć Č č | Ď ě Đ đ Ě ě Ğ ğ | Ģ ģ Ĩ ĩ     |
| IJ ij Ļ ļ Ľ ľ   | Ń ń Ņ ņ Œ œ     | Ř ř Ť ř     |
| Š š Š š Ţ ţ Ţ ţ | Ů ů Ű ű         | Ž ž Ž ž Ž ž |



Some of these characters expand differently, which can affect index sorting. For example, `ä` becomes `\IeC_L{"a}` and `Æ` becomes `\IeC_L{\AE_L}`. 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 Congrefs, July 4, 1776.”

```
\usepackage{newunicodechar}
\DeclareTextSymbolDefault{\textlongS}{TS1}
\DeclareTextSymbol{\textlongS}{TS1}{115}
\newunicodechar{f}{\textlongS}
```



Although `\newunicodechar{ā}{\=a}` allows `\Name{Ghazāli}` to show **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. Understandably, 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                 | <code>\foo abc\relax</code>    | <code>&lt;ab&gt;&lt;c&gt;</code> |
| <code>{æ}bc</code>  | <code>\foo {æ}bc\relax</code>  | <code>&lt;æb&gt;&lt;c&gt;</code> |
| <code>\ae bc</code> | <code>\foo \ae bc\relax</code> | <code>&lt;æb&gt;&lt;c&gt;</code> |

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                           |
|------------------|-----------------------------|-----------------------|----------------------------------|
| <code>æbc</code> | <code>\foo æbc\relax</code> | <code>xelatex</code>  | <code>&lt;æb&gt;&lt;c&gt;</code> |
| <code>æbc</code> | <code>\foo æbc\relax</code> | <code>lualatex</code> | <code>&lt;æb&gt;&lt;c&gt;</code> |
| <code>æbc</code> | <code>\foo æbc\relax</code> | <code>pdflatex</code> | <code>&lt;æ&gt;&lt;bc&gt;</code> |



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 67 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 test for equality. “Active” characters expand to “two-argument wide” values under NFSS, as the table below shows via defining a macro to be a character, then printing its `\meaning` in the cell:

| <code>\def\@a{&lt;L&gt;}</code> | <code>\protected@edef\@a{&lt;L&gt;}</code> | <code>\protected\edef\@a{&lt;L&gt;}</code>  |
|---------------------------------|--|---|
| <code>A macro:-&gt;A</code>     | <code>A macro:-&gt;A</code>                | <code>A \protected macro:-&gt;A</code>      |
| <code>Ã macro:-&gt;ÃÃ</code>    | <code>Ã macro:-&gt;\IeC {\‘A}</code>       | <code>Ã \protected macro:-&gt;Ã</code>      |
| <code>ß macro:-&gt;Ã§</code>    | <code>ß macro:-&gt;\IeC {\ss }</code>      | <code>ß \protected macro:-&gt;\T1\ss</code> |

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



This method of testing for active characters and resolving the related issues can interfere with some situations of expansion, generating errors. Be mindful of names within an `\edef`, for example, unless you control expansion explicitly.



L<sup>A</sup>T<sub>E</sub>X 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                            |
|--------------------|-------------------------------|-----------------------------------|
| <code>a b c</code> | <code>\foo a b c\relax</code> | <code>&lt;ab&gt;&lt; c&gt;</code> |
| <code>ab c</code>  | <code>\foo ab c\relax</code>  | <code>&lt;ab&gt;&lt; c&gt;</code> |
| <code>a bc</code>  | <code>\foo a bc\relax</code>  | <code>&lt;ab&gt;&lt;c&gt;</code>  |
| <code>abc</code>   | <code>\foo abc\relax</code>   | <code>&lt;ab&gt;&lt;c&gt;</code>  |

Using explicit spacing macros prevents gobbled spaces:

| Argument                       | Macro                                     | Result                            |
|--------------------------------|---|-----------------------------------|
| <code>a~bc</code>              | <code>\foo a~bc\relax</code>              | <code>&lt;a &gt;&lt;bc&gt;</code> |
| <code>a\nobreakspace bc</code> | <code>\foo a\nobreakspace bc\relax</code> | <code>&lt;a &gt;&lt;bc&gt;</code> |
| <code>a\space bc</code>        | <code>\foo a\space bc\relax</code>        | <code>&lt;a &gt;&lt;bc&gt;</code> |

See also Sections 2.3.6 and 2.3.7, as well as Section 2.5.4.

## 2.9.4 L<sup>A</sup>T<sub>E</sub>X Engines



The `nameauth` package tries to work with multiple languages and typesetting engines. The following preamble snippet illustrates how that can be done:<sup>28</sup> Please note that there have been changes to the `fontspec` package in 2016, so one should consult the relevant package documentation regarding the new encodings and what packages to include or not to include as a result.

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

This general arrangement works for this manual, which is tested with all of the L<sup>A</sup>T<sub>E</sub>X 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 <pdf latex text>%
    \else <latex text>%
    \fi
  \fi
\fi
```

---

<sup>28</sup>A similar version of this example is in `examples.tex`, collocated with this manual.

### 2.9.5 Hooks: Intro

Starting with this section we reset all formatting hooks to do nothing. This helps us focus on the modifications made hereafter.

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:<sup>29</sup>

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

Changes to `\NamesFormat` are not relying just on scoping rules to keep them “local.” We use `\let` to make explicit changes in order to avoid some possible side effects. We now use the example above in a sample text:

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

\Name{Vlad III, Dracula}, known as \AKA{Vlad III, Dracula}{Vlad,
Țepeș} (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 Țepeș (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 default `\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 any of the formatting hooks, 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.

<sup>29</sup>A similar version of this example is in `examples.tex`, collocated with this manual.

## 2.9.6 Hooks: Life Dates

We can use name conditionals (Section 2.7.1) and text tags (Section 2.6) 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 of `\AKA`. We also use a different formatting for the naming macros on the one hand and `\AKA` on the other:<sup>30</sup>

```
\newif\ifNoTag%           allows us to work around \ForgetName
\let\OldFormat\NamesFormat%           save the format
\let\OldFrontFormat\FrontNamesFormat
\makeatletter%           access internals
\renewcommand*\NamesFormat[1]{\begingroup%
  \protected@edef\temp{\endgroup\textbf{#1}}%
  \unless\ifNoTag
    \if@nameauth@InName
      {\bfseries\noexpand\NameQueryInfo
        [\unexpanded\expandafter{\the\@nameauth@toksa}]
        {\unexpanded\expandafter{\the\@nameauth@toksb}}
        [\unexpanded\expandafter{\the\@nameauth@toksc}]]\fi
    \if@nameauth@InAKA\noexpand\NameQueryInfo
      [\unexpanded\expandafter{\the\@nameauth@toksa}]
      {\unexpanded\expandafter{\the\@nameauth@toksb}}
      [\unexpanded\expandafter{\the\@nameauth@toksc}]]\fi
  \fi}\temp\global\NoTagfalse%
}
\makeatother
\let\FrontNamesFormat\NamesFormat
```

This change prints tags in the first use hooks unless `\NoTag` is set true. Please note that the conditional path here is placed within the `\edef`. Putting it outside the `\edef`, such as `\unless\ifNoTag\temp\fi`, will cause errors.

This method uses the  $\epsilon$ -TeX primitives `\noexpand` and `\unexpanded` to avoid the extensive repetition of `\expandafter`. Since the `nameauth` package depends on `etoolbox`, we assume that we are using  $\epsilon$ -TeX.

Before we can refer to any text tags, we must create them. Using the approach above, we must include a leading space in the text tags:

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

<sup>30</sup>A similar version of this example is in `examples.tex`, collocated with this manual.

The leading space is needed only when a text tag appears. Another way to add that space is to put it in the conditional path of the formatting hook and leave it out of the text tags entirely:

```
... \unless\ifNoTag...{ }\noexpand\NameQueryInfo...\fi}\temp
```

Now we begin with the first example, where both the name and the dates are in boldface because we use a naming macro:

```
\ForgetThis\Wash held office 1789--97. No tags here: \Wash.
```

```
First use, dates suppressed: \NoTagtrue\ForgetThis\Wash.
```

```
George Washington (1732–99) held office 1789–97. No tags here:
```

```
Washington. First use, dates suppressed: George Washington.
```

Since \AKA usually calls the “subsequent use” formatting hooks, we can create a scope to “fool” it into calling the first-use hook via \let:

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

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



Another solution uses the `formatAKA` package option. In the example below, we simulate a first occurrence of Kemal. Then we simulate `formatAKA`. Finally, we use `\ForceName` with `\AKA`:

```
\ForgetName[Mustafa]{Kemal}% first use
\makeatletter\@nameauth@AKAFormattrue\makeatother% formatAKA
\Name[Mustafa]{Kemal} was granted the name%
\AKA[Mustafa]{Kemal}{Atatürk}. We mention%
\AKA[Mustafa]{Kemal}{Atatürk} again.
```

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

There are other solutions for getting this result, such as using `\IncludeName*` or non-printing control sequences. One must decide the best approach for oneself. Please remember to reset the formatting, if needed:

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

See Section 3.4 and page 79 for the decision paths and the logic used by the package. Presently, writing hook macros is much simpler.

Back to Section 1.6

## 2.9.7 Hooks: Advanced

### Alternate Formatting

- 3.1** The alternate formatting framework now makes designing hooks much easier by providing some built-in features that add not only error protection but also ease of use. We enabled that framework at the beginning of this section with `\AltFormatActive` and take care not to use the names in this section elsewhere.

Both `\AltFormatActive` and `\AltFormatActive*` set the internal Boolean flag `\@nameauth@AltFormattrue`, which enables alternate formatting. Additionally, `\AltFormatActive` sets `\@nameauth@DoAlttrue`, which “switches on” alternate formatting. `\AltFormatInactive` sets both flags false.

`\CapThis` protection The main feature of this framework is protecting against errors created when `\@nameauth@Cap` gets a misleading result from `\@nameauthUTFtest` and splits a token list in a way that causes an error. The alternate capping macro `\AltCaps` and `\CapThis` work mutually in `\@nameauth@Parse` to ensure that they do not interfere with each other, as we saw demonstrated in Section 2.4.3.

### Continental Format

Here we look in greater detail at the more complex version of Continental formatting from Section 2.4.3.

changes in text Font changes in the text occur with the short macros `\textSC`, `\textIT`, `\textBF`, and `\textUC`. They all look similar to `\textSC`. We therefore show just this one macro as an example from the package source.

```
\newcommand*\textSC[1]{%
  \if@nameauth@DoAlt\textsc{#1}\else#1\fi
}
```

Using this method, formatting occurs in both the text and in the index if the `altformat` option or `\AltFormatActive` was used. If you use a name that uses these macros both within and outside of the alternate formatting regime, you will get spurious index entries.<sup>31</sup>

We plan to have small caps on by default, then off in subsequent uses. We thus use `\AltFormatActive` for the “always on” general condition, then redefine `\MainNameHook` because it is the subsequent use. We use `\AltOff` to suppress formatting. It works only in the formatting hooks. `\AltOff` toggles an internal flag that deactivates any changes. From the source, it looks like:

```
\newcommand*\AltOff{%
  \if@nameauth@InHook\@nameauth@DoAltfalse\fi
}
```

Since the normal effects of `\CapThis` are disabled, `\AltCaps` does the job by capitalizing its argument in braces `{ }` when it is used in a macro hook and triggered by `\CapThis`. The source looks like:

---

<sup>31</sup>Using `\AltFormatActive*` is interesting because it looks like the normal `nameauth` regime but prevents `\CapThis` from having its normal effect unless you use `\AltCaps`. With `\AltFormatActive*` if you use a name that has alternate formatting both within and outside of the alternate formatting regime, you may not get spurious index entries as long as control sequences are consistent.

```

\newcommand*\AltCaps[1]{%
  \if@nameauth@InHook
    \if@nameauth@DoCaps\uppercase{#1}\else#1\fi
  \else#1\fi
}

```

It is important that these macros not expand too soon. We therefore must put `\noexpand` once before `\textSC`, etc., and once before `\AltCaps`. This is because the name arguments in `nameauth` have to use `\protected@edef` to work right. We will get to that when we set up the names and any applicable tags.

Before we alter the formatting hooks, we can save the hook macros if we want to recall them (below) or we can use `\begingroup` and `\endgroup` to create a new scope and let that handle any changes. We use scoping in this section.

The final step *does not come* from the `nameauth` source. We must redefine the formatting hooks ourselves. One of the simplest ways to do this when using the `altformat` option or `\AltFormatActive` is:

```

\renewcommand*\MainNameHook{\AltOff}

```

Simple, *oder?* If needed, we can `\let\FrontNameHook\MainNameHook`. If you want to suppress formatting altogether in the front matter, make the following change: `\let\FrontNamesFormat\MainNameHook`.

Continental formatting usually alters at least one element in the required name field, as we see below:

```

\begin{nameauth}
  \< Adams    & John  & \noexpand\textSC{Adams}      & >
  \< SDJR     & Sammy & \noexpand\textSC{Davis},
                                \noexpand\textSC{Jr}.    & >
  \< HAR      &      & Harun, \noexpand\textSC%
                                {\noexpand\AltCaps{a}l-Rashid} & >
  \< Mencius  &      & \noexpand\textSC{Mencius}      & >
\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]{\noexpand\textSC{Adams}}{Adams, John}
\PretagName[Sammy]%
  {\noexpand\textSC{Davis}, \noexpand\textSC{Jr}.}%
  {Davis, Sammy, Jr.}
\PretagName{Harun, \noexpand\textSC%
  {\noexpand\AltCaps{a}l-Rashid}}{Harun al-Rashid}
\PretagName{\noexpand\textSC{Mencius}}{Mencius}

```

The use in the body text is not much different than normal, but only if we use the simplified interface.

| First           | Next    | Long            | Short   |
|-----------------|---------|-----------------|---------|
| John ADAMS      | Adams   | John Adams      | John    |
| Sammy DAVIS JR. | Davis   | Sammy DAVIS JR. | Sammy   |
| Harun AL-RASHID | Harun   | Harun al-Rashid | Harun   |
| MENCIUS         | Mencius | Mencius         | Mencius |

- Punctuation detection works: Sammy DAVIS JR. Then we have Davis.
- `\DropAffix\LSDJR` gives Sammy Davis.
- `\RevComma\LAdams` yields Adams, John. All the reversing macros work.
- `\ForceName\ForceFN\SHAR` produces AL-RASHID. If we add `\CapThis` we get AL-RASHID. The way that Continental resources treat certain affixes relates to similar issues in [Mulvany, 168–73].<sup>32</sup>
- One must include the extra control sequences in all the macro arguments that use these names.

If we use the `formatAKA` option we can refer to Mencius as MENG Ke, and again Meng Ke. We get that with:

```
\PretagName{\noexpand\textSC{Meng}, Ke}{Meng Ke}
\AKA{\noexpand\textSC{Mencius}}{\noexpand\textSC{Meng}, Ke}
```

## Rolling Your Own: New Style

“New style” means that we are sticking closely with various package features that have been implemented already and look similar to the solutions in Section 2.4.3. Here we set out on the path to custom formatting.



When redesigning formatting hooks, you should use `\AltFormatActive` or the `altformat` option to enable alternate formatting and prevent `\CapThis` from breaking your formatting macros.

We recommend using the internal package flag `\@nameauth@DoAlt`, which activates alternate formatting, `\@nameauth@DoCaps`, which handles capitalization, and `\@nameauth@InHook`, which is true when the formatting hooks are called. See page 77 and following for examples of how the ready-made formatting macros are designed. If you create your own macros, they will look similar:<sup>33</sup>

```
\makeatletter%
\newcommand*\Fbox[1]{%
  \if@nameauth@DoAlt\fbox{#1}\else#1\fi
}
\makeatother
```

Since `\AltCaps` is part of `nameauth`, you need not reinvent that particular wheel. As was the case previously, the final step is redefining the formatting hooks. One of the simplest ways to do this is:

```
\renewcommand*\MainNameHook{\AltOff}
\let\FrontNameHook\MainNameHook
```

When defining names, be sure to use `\noexpand` before the control sequences in the macro arguments so they expand at the proper time:

```
\PretagName[Pierre-Jean]%
{\noexpand\Fbox{\noexpand\AltCaps{d}e Smet}}%
{de Smet, Pierre-Jean}
```

<sup>32</sup>Handling non-Western names in Western sources can be a gray area. One ought take care to be culturally sensitive in these matters.

<sup>33</sup>A similar version of this example is in `examples.tex`, collocated with this manual.



```

\begin{nameauth}
  \< deSmet & Pierre-Jean &
      \noexpand\Fbox{\noexpand\AltCaps{d}e Smet}} & >
\end{nameauth}

```

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

| First   | Next                 | Long                  | Short                 |
|---|----------------------|-----------------------|-----------------------|
| <code>\deSmet</code>  | <code>\deSmet</code> | <code>\LdeSmet</code> | <code>\SdeSmet</code> |
| Pierre-Jean <span style="border: 1px solid black; padding: 0 2px;">de Smet</span> | de Smet              | Pierre-Jean de Smet   | Pierre-Jean           |

The capitalized version `\CapThis\deSmet` is De Smet. This also works for a formatted use via `\ForceName:` 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.

## Rolling Your Own: Old Style

The idea of “old style” conveys the meaning that these approaches have something of an independent design that goes beyond what the package offers. Since older `nameauth` versions had less integration of Continental formatting, we use this parlance. Here we start to use more of our own solutions for dealing with name formatting. We begin that journey by looking at `\NameParser`.

`\NameParser`  
3.1

This user-accessible parser (page 79) builds a name from the internal macros `\FNN`, `\SNN`, `\rootb` and `\suffb`. Reversing and commas are still usable; capitalization depends on the context. The general form is:

```
\renewcommand*{Hook}[1]{...\NameParser...}
```

In order to use this hook-level parser, we want the option of ignoring the text that is sent to the formatting hooks from `\@nameauth@Parse`. We do that by redefining the hooks to take an argument.

If we use the `altformat` option or `\AltFormatActive`, then alternate formatting is both enabled and “switched on”; whatever formatting macros that we are using should be in the “on” state. If we want subsequent uses of names to be in the “off” state, we can design a hook like:

```
\renewcommand*{Hook}[1]{...\AltOff\NameParser...}
```

If we used `\AltFormatActive*`, where the formatting macros are “switched off” but enabled nonetheless, then we might want a hook that turns the macros “on” instead:

```
\renewcommand*{Hook}[1]{...\AltOn\NameParser...}
```



We have shown already that you do not really need `\NameParser` to use these switching macros in the hooks. Yet the user-level parser does have some handy uses, especially as we go further toward designing custom macros. For example, we demonstrate an extreme case based on Section 2.9.5 where we modify some internal flags to have `\NameParser` to produce different syntactic forms than the normal output (index entries suppressed):<sup>34</sup>

<sup>34</sup>A similar version of this example is in `examples.tex`, collocated with this manual.

```

\makeatletter
\renewcommand*\NamesFormat[1]{#1\unless\ifinner
  \marginpar{\small\raggedleft%
    \@nameauth@FullNametrue\@nameauth@FirstNamefalse%
    \@nameauth@EastFNfalse\NameParser}\fi}
\renewcommand*\MainNameHook[1]{\AltOff#1\unless\ifinner
  \marginpar{\small\raggedleft%
    \@nameauth@FullNamefalse\@nameauth@FirstNamefalse%
    \@nameauth@EastFNfalse\NameParser}\fi}
\makeatother

Wm. SHAKESPEARE      Wm. SHAKESPEARE      \Name[Wm.]{\noexpand\textSC{Shakespeare}}
Shakespeare          Shakespeare          \Name[Wm.]{\noexpand\textSC{Shakespeare}}
Shakespeare          Wm. Shakespeare      \Name*[Wm.]{\noexpand\textSC{Shakespeare}}
Shakespeare          William              \FName[Wm.]{\noexpand\textSC{Shakespeare}}[William]
Wm. SHAKESPEARE      SHAKESPEARE
\ForceName\Name[Wm.]{\noexpand\textSC{Shakespeare}}

```

In a first-use hook, the person’s full name always is displayed in the margin. In a subsequent-use formatting hook, only a surname, ancient personal name, or mononym can be displayed in the margin.

We use the `\NameParser` macro to re-create the name, but using different rules via the internal Boolean flags. The macros that toggle these flags are discussed elsewhere. These include:

|  |                                    |
|--|------------------------------------|
| <code>\if@nameauth@FullName</code>   | Print a full name if true.         |
| <code>\if@nameauth@FirstName</code>  | Print a first name if true.        |
| Only one or the other of these can be true to avoid undocumented behavior. |                                    |
| <code>\if@nameauth@RevThis</code>  | Reverse name order if true.        |
| <code>\if@nameauth@EastFN</code>   | toggled by <code>\ForceFN</code> . |
| <code>\if@nameauth@RevThisComma</code>                                     | Reverse Western name, add comma.   |
| Reversing without commas overrides reversing with commas.                  |                                    |



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

The older version of “rolling your own” is reminiscent of the newer way, but it has significant differences:

- We do not use the internal package macros.
- We best use `\NameParser` to generate the name in the hooks. It may be possible not to do so, but as we get more customized the user-level parser is a handy way to get reasonably predictable results.
- We still recommend using `\AltFormatActive` if you want to disable the normal effects of `\CapThis`. Otherwise redefine `\CapThis` (which is what we do below).<sup>35</sup>

We define three Boolean flags and set one of them true by default. The `\iffbox` flag takes over the internal function of `\@nameauth@DoAlt`, which is

<sup>35</sup>A similar version of this example is in `examples.tex`, collocated with this manual.

enabled by `\AltFormatActive`. The `\ifFirstCap` flag takes over the internal function of `\@nameauth@DoCaps`, which is enabled by `\CapThis`. The `\ifInHook` flag replaces the internal function of `\@nameauth@InHook`, which is enabled by the internal format hook dispatcher.

```
\newif\ifFbox
\newif\ifFirstCap
\newif\ifInHook
\Fboxtrue
```

The formatting macro is like the new style, except it refers to `\ifFbox`:

```
\renewcommand*\Fbox[1]{%
  \ifFbox\fbbox{#1}\else#1\fi
}
```

Our new `\AltCaps` works like the built-in version, except it does not use the internal macros and flags:

```
\renewcommand*\AltCaps[1]{%
  \ifInHook
    \ifFirstCap\uppercase{#1}\else#1\fi
  \else
    #1%
  \fi
}
```

Here we redefine `\CapThis` to use our flag instead of the internal flag:

```
\renewcommand*\CapThis{\FirstCaptrue}
```

We have to do in our own hooks what the naming macros do internally in order to get the same exit conditions. In the new style, we do not have to define `\NamesFormat`. Here we have to define everything:

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

Instead of using just `\AltOff` before `\NameParser` below, we have to add a few extras in order to mimic the functions of the internal flags:

```
\renewcommand*\MainNameHook[1]
{%
  \Fboxfalse\InHooktrue\NameParser\InHookfalse%
  \global\FirstCapfalse\Fboxtrue%
}
```

We let the front-matter hooks be like the main-matter hooks to avoid spurious index entries.

```
\let\FrontNamesFormat\Namesformat
\let\FrontNameHook\MainNameHook
```

Because we uses `\noexpand`, our “old-style” macros will index the name below under the same entry as the “new-style” macros.

| First   | Next                 | Long                  | Short                 |
|---|----------------------|-----------------------|-----------------------|
| <code>\deSmet</code>  | <code>\deSmet</code> | <code>\LdeSmet</code> | <code>\SdeSmet</code> |
| Pierre-Jean <span style="border: 1px solid black; padding: 0 2px;">de Smet</span> | de Smet              | Pierre-Jean de Smet   | Pierre-Jean           |

The capitalized version `\CapThis\deSmet` is De Smet. This also works for a formatted use via `\ForceName: De Smet`.

Be sure not to use names with alternate formatting outside of those sections, or else you will get spurious index entries. We now resume normal formatting with `\AltFormatInactive`.

Back to Section 1.6

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

```
\makeatletter
\newcommand*{\MyName}[3][1=\@empty, 3=\@empty]{\langle Name \rangle}
\newcommand*{\MyLName}[3][1=\@empty, 3=\@empty]
{\langle Long name \rangle \@nameauth@FullName false}
\newcommand*{\MyFName}[3][1=\@empty, 3=\@empty]
{\langle Short name \rangle \@nameauth@FirstName false}
\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 \langle Name \rangle, \langle Long name \rangle, and \langle Short name \rangle.
```

`\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.

## 2.10 Technical Notes

About the package itself:

- We put great weight on being backward-compatible with older versions.
- Recent changes aim for simpler work flow, not more features.
- The package works with both `xindy` and `makeindex`. We recommend `xindy` for languages whose collating sequences do not map to English.<sup>36</sup>
- 3.0 • We support alternate names in both Western and “native” Eastern forms. Mononyms and the older syntax for non-Western names do not support alternate names.
- 3.0 • Name output, index entries, and index cross-references are independent modules.
- 3.0 • Warnings for the indexing macros are suppressed unless one uses the `verbose` option. The `nameauth` environment will continue to emit warnings as needed.
- 2.6 • The `comma` option and the older syntax are no longer restrictive, save with `\AKA` and its derivatives. See Sections 1.5, 2.3.1, and 2.8.
- 2.5 • No formatting is selected by default. Cf. Sections 2.4.2, 2.9.5, 2.9.6, and 2.9.7.

About the manual:

- This manual is compatible with both A4 and US letter formats.
- For an index that focuses on using the names, we minimize macro references.
- We mention when this manual changes package internals for an example.
- The name pattern reference was removed for redundancy and obsolescence.

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.<sup>37</sup>
- This package was built with `pdflatex`. This item changes per  $\text{\LaTeX}$  engine.
- This package is tested on Ubuntu Linux and Windows 7 (both vanilla  $\text{\TeX}$  Live). Cygwin provides `make` on Windows. The `pdflatex` version of this package is released from the Ubuntu platform to CTAN.

---

<sup>36</sup>`\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.

<sup>37</sup>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.

## 2.11 Errors and Warnings

Here are some ways to avoid common errors:

- Keep it simple! Avoid unneeded macros and use the simplified interface.
- Check braces and brackets with naming macros to avoid errors like “Paragraph ended...” and “Missing *⟨grouping token⟩* inserted.”
- Do not apply a formatting macro to an entire comma-delimited *⟨SNN, affix⟩* pair. Format each part separately.
- Consider using `\PretagName` with all names containing control sequences or active Unicode; see Section 2.5.4.
- One way to spot errors is to compare index entries with names in the body text. All macros that produce output also emit meaningful warnings.

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 2.8.

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 happens 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`, `\AKA`, or `\IndexRef`, respectively. Those macros set these flags. On their use, see also Sections 2.9.6 and 2.9.7.

```
1 \newif\if@nameauth@InAKA
2 \newif\if@nameauth@InName
3 \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 2.9.6 and 2.9.7.

```
4 \newif\if@nameauth@Lock
5 \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 `\IndexInactive` or the `index` and `noindex` options toggle the first flag; `\SkipIndex` toggles the second. `\JustIndex` toggles the third, which makes the core naming engine act like a call to `\IndexName`:

```
6 \newif\if@nameauth@DoIndex
7 \newif\if@nameauth@SkipIndex
8 \newif\if@nameauth@JustIndex
```

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

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

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

```
10 \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.

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

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

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

These two flags trigger `\ForgetName` and `\SubvertName` within `\@nameauth@Name`.

```
14 \newif\if@nameauth@Forget
15 \newif\if@nameauth@Subvert
```

`\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.

```
16 \newif\if@nameauth@FirstFormat
17 \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.

```
18 \newif\if@nameauth@AlwaysComma
19 \newif\if@nameauth@ShowComma
20 \newif\if@nameauth@NoComma
```

## Name Breaking

`\KeepAffix` toggles the first flag below, while `\KeepName` toggles the second. Both affect the use of non-breaking spaces in the text.

```
21 \newif\if@nameauth@NBSP
22 \newif\if@nameauth@NBSPX
```

## Detect Punctuation

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

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

## Long and Short Names

`\if@nameauth@FullName` is true for a long name reference. `\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.

```
24 \newif\if@nameauth@FullName
25 \newif\if@nameauth@FirstName
26 \newif\if@nameauth@AltAKA
27 \newif\if@nameauth@OldAKA
28 \newif\if@nameauth@ShortSNN
29 \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.

```
30 \newif\if@nameauth@RevAll
31 \newif\if@nameauth@RevThis
32 \newif\if@nameauth@AllCaps
33 \newif\if@nameauth@AllThis
```

## Last-Comma-First

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

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

## Capitalize First Letter

The next 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. The fourth prevents the first-letter capping mechanism from interacting with Continental formatting and the fifth toggles it.

```
36 \newif\if@nameauth@DoCaps
37 \newif\if@nameauth@UTF
```



```

38 \newif\if@nameauth@Accent
39 \newif\if@nameauth@AltFormat
40 \newif\if@nameauth@DoAlt

```

## 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.

```
41 \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`.

```

42 \newtoks\@nameauth@toksa%
43 \newtoks\@nameauth@toksb%
44 \newtoks\@nameauth@toksc%

```

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

```

45 \newtoks\@nameauth@etoksb%
46 \newtoks\@nameauth@etoksc%
47 \newtoks\@nameauth@etoksd%

```

## 3.2 Hooks

`\NamesFormat` Post-process “first” instance of final complete name form in text. See Sections 2.4.2 and 2.9.5f. Called when both `\@nameauth@MainFormat` and `\@nameauth@FirstFormat` are true.

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

`\MainNameHook` Post-process subsequent instance of final complete name form in main-matter text. See Sections 2.4.2 and 2.9.5f. Called when `\@nameauth@MainFormat` is true and the Boolean flag `\@nameauth@FirstFormat` is false.

```
49 \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.

```
50 \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.

```
51 \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 2.9.8. Use at your own risk! Changing these macros basically rewrites this package.

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

`\NameauthLName` Customization hook called after `\@nameauth@FullName` is set true. See Section 2.9.8.

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

`\NameauthFName` Customization hook called after `\@nameauth@FirstName` is set true. See Section 2.9.8.

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

### 3.3 Package Options

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

```

55 \DeclareOption{comma}{\@nameauth@AlwaysCommatrue}
56 \DeclareOption{nocomma}{\@nameauth@AlwaysCommafalse}
57 \DeclareOption{mainmatter}{\@nameauth@MainFormattrue}
58 \DeclareOption{frontmatter}{\@nameauth@MainFormatfalse}
59 \DeclareOption{formatAKA}{\@nameauth@AKAFormattrue}
60 \DeclareOption{oldAKA}{\@nameauth@OldAKAtrue}
61 \DeclareOption{index}{\@nameauth@DoIndextrue}
62 \DeclareOption{noindex}{\@nameauth@DoIndexfalse}
63 \DeclareOption{pretag}{\@nameauth@Pretagtrue}
64 \DeclareOption{nopretag}{\@nameauth@Pretagfalse}
65 \DeclareOption{allcaps}{\@nameauth@AllCapstrue}
66 \DeclareOption{normalcaps}{\@nameauth@AllCapsfalse}
67 \DeclareOption{allreversed}%
68   {\@nameauth@RevAlltrue\@nameauth@RevAllCommafalse}
69 \DeclareOption{allrevcomma}%
70   {\@nameauth@RevAllfalse\@nameauth@RevAllCommatrue}
71 \DeclareOption{notreversed}%
72   {\@nameauth@RevAllfalse\@nameauth@RevAllCommafalse}
73 \DeclareOption{alwaysformat}{\@nameauth@AlwaysFormattrue}
74 \DeclareOption{smallcaps}{\renewcommand*\NamesFormat{\scshape}}
75 \DeclareOption{italic}{\renewcommand*\NamesFormat{\itshape}}
76 \DeclareOption{boldface}{\renewcommand*\NamesFormat{\bfseries}}
77 \DeclareOption{noformat}{\renewcommand*\NamesFormat{}}
78 \DeclareOption{verbose}{\@nameauth@Verbosetrue}
79 \DeclareOption{altformat}{\@nameauth@AltFormattrue\@nameauth@DoAlttrue}
80 \ExecuteOptions%
81   {nocomma,mainmatter,index,pretag,%
82     normalcaps,notreversed,noformat}
83 \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.

```

84 \RequirePackage{etoolbox}
85 \RequirePackage{suffix}
86 \RequirePackage{trimspaces}
87 \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.

```

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

```

#### Parsing: Root and Suffix

`\@nameauth@Root` The following two macros return everything before a comma in  $\langle SNN \rangle$ .

```

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

```

`\@nameauth@TrimRoot` Throw out the comma and suffix, return the radix.  
91 `\def\@nameauth@TrimRoot#1,#2\\{\trim@spaces{#1}}`

`\@nameauth@TagRoot` The following two macros return everything before a vertical bar (|) in an index tag.  
92 `\newcommand*\@nameauth@TagRoot[1]{\@nameauth@TrimTag#1|\\}`

`\@nameauth@TrimTag` Throw out the bar and suffix, return the radix.  
93 `\def\@nameauth@TrimTag#1|#2\\{#1}`

`\@nameauth@Suffix` The following two macros parse  $\langle SNN \rangle$  into a radix and a comma-delimited suffix, returning only the suffix after a comma in the argument, or nothing.  
94 `\newcommand*\@nameauth@Suffix[1]{\@nameauth@TrimSuffix#1,,\\}`

`\@nameauth@TrimSuffix` Throw out the radix; return the suffix with no leading spaces.  
95 `\def\@nameauth@TrimSuffix#1,#2,#3\\%`  
96 `{\ifx\\#2\\ \@empty\else\trim@spaces{#2}\fi}`

## Parsing: Capitalization

`\@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 in 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`.

```

97 \newcommand*\@nameauth@UTFtest[1]
98 {%
99   \ifdefined\Umathchar
100     \@nameauth@UTFfalse%
101   \else
102     \ifdefined\UTFviii@two@octets
103       \if@nameauth@Accent
104         \@nameauth@UTFtrue\@nameauth@Accentfalse%
105       \else
106         \toks@ \expandafter{\@car#1\@nil}%
107         \edef\one{\the\toks@}%
108         \toks@ \expandafter{\@car\@nil}%
109         \edef\two{\the\toks@}%
110         \ifx\one\two\@nameauth@UTFtrue\else\@nameauth@UTFfalse\fi
111       \fi
112     \else
113       \@nameauth@UTFfalse%
114     \fi
115   \fi
116 }
```

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

```

117 \newcommand*\@nameauth@Cap[1]
118 {%
119   \if@nameauth@UTF \@nameauth@Ciii#1\\%
120   \else \@nameauth@Cii#1\\%
121   \fi
122 }
```

`\@nameauth@Cii` Cap the first letter of the argument.

```

123 \def\@nameauth@Cii#1#2\\%
124 {\expandafter\trim@spaces\expandafter{\uppercase{#1}#2}}
```

`\@nameauth@Ciii` Cap the first letter of the argument. This is called in pdf<sub>l</sub>atex under inputenc/Unicode.

```
125 \def\@nameauth@Ciii#1#2#3\\%
126 {\expandafter\trim@spaces\expandafter{\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.

```
127 \newcommand*\@nameauth@TestDot[1]
128 {%
129   \def\TestDot##1.\TestEnd##2\\{\TestPunct{##2}}%
130   \def\TestPunct##1{%
131     \ifx\TestPunct##1\TestPunct%
132     \else
133       \@nameauth@Puncttrue%
134     \fi
135   }%
136   \@nameauth@Punctfalse%
137   \TestDot#1\TestEnd.\TestEnd\\%
138 }
```

`\@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.

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

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

```
141 \newcommand*\@nameauth@EvalDot%
142 {%
143   \let\@period=.%
144   \ifx\@token\@period\expandafter\@gobble \fi
145 }
```

## 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 an empty root within a root/suffix pair.

```
146 \newcommand*\@nameauth@Error[2]
147 {%
148   \edef\msg{#2 SNN field empty}%
149   \edef\msgb{#2 SNN field malformed}%
150   \protected@edef\testname{\trim@spaces{#1}}%
151   \protected@edef\testroot{\@nameauth@Root{#1}}%
152   \ifx\testname\@empty
153     \PackageError{nameauth}{\msg}%
154   \fi
155   \ifx\testroot\@empty
156     \PackageError{nameauth}{\msgb}%
157   \fi
158 }
```

## 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.

```
159 \newcommandx*\@nameauth@Name[3][1=\@empty, 3=\@empty]
160 {%
```

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

```
161 \unless\if@nameauth@Lock
162 \@nameauth@Locktrue%
```

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

```
163 \@nameauth@InName>true%
164 \@nameauth@Error{#2}{\macro \string\@nameauth@name}%
```

If we use `\JustIndex` then skip everything else..

```
165 \if@nameauth@JustIndex
166 \IndexName[#1]{#2}[#3]%
167 \@nameauth@InName>false%
168 \@nameauth@Lock=false%
169 \@nameauth@JustIndex=false%
170 \else
```

Delete/create name cseq if directed. If the delete flag is set, the create flag is ignored. Ensure that names are printed in horizontal mode. Print the name between two index entries, if allowed.

```
171 \if@nameauth@Forget
172 \ForgetName[#1]{#2}[#3]%
173 \else
174 \if@nameauth@Subvert
175 \SubvertName[#1]{#2}[#3]%
176 \fi
177 \fi
178 \leavevmode\hbox{}%
179 \unless\if@nameauth@SkipIndex\IndexName[#1]{#2}[#3]\fi
180 \if@nameauth@MainFormat
181 \@nameauth@Parse[#1]{#2}[#3]{!MN}%
182 \else
183 \@nameauth@Parse[#1]{#2}[#3]{!NF}%
184 \fi
185 \unless\if@nameauth@SkipIndex\IndexName[#1]{#2}[#3]\fi
```

Reset all the “per name” Boolean values.

```
186 \@nameauth@SkipIndex=false%
187 \@nameauth@Forget=false%
188 \@nameauth@Subvert=false%
189 \@nameauth@Lock=false%
190 \@nameauth@InName=false%
191 \@nameauth@NBSP=false%
192 \@nameauth@NBSPX=false%
193 \@nameauth@DoCaps=false%
194 \@nameauth@Accent=false%
195 \@nameauth@AllThis=false%
196 \@nameauth@ShowComma=false%
197 \@nameauth@NoComma=false%
198 \@nameauth@RevThis=false%
```

```

199      \@nameauth@RevThisCommafalse%
200      \@nameauth@ShortSNNfalse%
201      \@nameauth@EastFNfalse%
202      \fi

```

Close the “locked” branch.

```
203      \fi
```

Call the full stop detection.

```

204      \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
205  }

```

`\@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.

```

206 \newcommandx*\@nameauth@Parse[4][1=\@empty, 3=\@empty]
207 {%
208   \if@nameauth@Lock
209     \let\ex\expandafter%

```

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

```

210     \protected@edef\arga{\trim@spaces{#1}}%
211     \protected@edef\rootb{\@nameauth@Root{#2}}%
212     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
213     \protected@edef\argc{\trim@spaces{#3}}%

```

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

```

214     \if@nameauth@AllCaps\@nameauth@AllThistrue\fi
215     \if@nameauth@RevAll\@nameauth@RevThistrue\fi
216     \if@nameauth@RevAllComma\@nameauth@RevThisCommatrue\fi

```

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

```

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

```

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

```

220     \@nameauth@toksa\expandafter{#1}%
221     \@nameauth@toksb\expandafter{#2}%
222     \@nameauth@toksc\expandafter{#3}%

```

Implement capitalization on demand in the body text if not in Continental mode.

```

223     \if@nameauth@DoCaps
224       \let\csuffb\suffb%
225       \let\cargc\argc%
226       \unless\if@nameauth@AltFormat

```

We test the root surname for active Unicode characters. Then we capitalize the first letter.

```

227         \ex\@nameauth@UTFtest\ex{\rootb}%
228         \edef\crootb{\noexpand\@nameauth@Cap{\rootb}}%

```

We test the suffix for active Unicode characters. Then we capitalize the first letter.

```

229         \unless\ifx\suffb\@empty
230           \ex\@nameauth@UTFtest\ex{\suffb}%
231           \edef\csuffb{\noexpand\@nameauth@Cap{\suffb}}%
232         \fi

```

We test the final optarg for active Unicode characters. Then we capitalize the first letter.

```

233      \unless\ifx\argc\@empty
234      \ex\@nameauth@UTFtest\ex{\argc}%
235      \edef\cargc{\noexpand\@nameauth@Cap{\argc}}%
236      \fi
237      \let\rootb\crootb%
238      \fi
239      \else

```

We capitalize the entire surname when desired; different from above.

```

240      \if@nameauth@AllThis
241      \protected@edef\rootb{\uppercase{\@nameauth@Root{#2}}}%
242      \fi
243      \fi

```

Use non-breaking spaces and commas as desired.

```

244      \edef\Space{\space}%
245      \edef\SpaceX{\space}%
246      \if@nameauth@NBSP\edef\Space{\nobreakspace}\fi
247      \if@nameauth@NBSPX\edef\SpaceX{\nobreakspace}\fi
248      \unless\ifx\arga\@empty
249      \if@nameauth@AlwaysComma
250      \edef\Space{,\space}%
251      \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
252      \fi
253      \if@nameauth@ShowComma
254      \edef\Space{,\space}%
255      \if@nameauth@NBSP\edef\Space{,\nobreakspace}\fi
256      \fi
257      \if@nameauth@NoComma
258      \edef\Space{\space}%
259      \if@nameauth@NBSP\edef\Space{\nobreakspace}\fi
260      \fi
261      \fi

```

We parse names by attaching “meaning” to patterns of macro arguments primarily via \FNN and \SNN. Then we call the name printing macros, based on the optional arguments.

```

262      \let\SNN\rootb%
263      \ifx\arga\@empty
264      \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.

```

265      \let\FNN\suffb%
266      \unless\ifx\suffb\@empty
267      \if@nameauth@DoCaps\let\FNN\csuffb\fi
268      \fi
269      \let\SNN\rootb%
270      \@nameauth@NonWest{\csb#4}%
271      \else

```

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

```

272      \ifx\suffb\@empty
273      \if@nameauth@DoCaps\let\FNN\cargc\else\let\FNN\argc\fi
274      \let\SNN\rootb%

```

```

275         \@nameauth@NonWest{\csbc#4}%
276     \else
277         \if@nameauth@DoCaps\let\FNN\cargc\else\let\FNN\argc\fi
278         \let\SNN\rootb%
279         \@nameauth@NonWest{\csb#4}%
280     \fi
281 \fi
282 \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.

```

283     \ifx\argc\@empty
284         \let\FNN\arga%
285     \else
286         \let\FNN\argc%
287     \fi
288     \unless\ifx\suffb\@empty
289         \protected@edef\SNN{\rootb\Space\suffb}%
290         \if@nameauth@ShortSNN\let\SNN\rootb\fi
291     \fi
292     \@nameauth@West{\csab#4}%
293 \fi
294 \fi
295 }

```

`\@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.

```

296 \newcommand*\@nameauth@NonWest[1]
297 {%
298     \if@nameauth@Lock
299         \unless\ifcsname#1\endcsname
300             \@nameauth@FirstFormattrue%
301         \fi
302     \if@nameauth@InAKA
303         \if@nameauth@AltAKA
304             \if@nameauth@OldAKA\@nameauth@EastFNtrue\fi
305             \@nameauth@FullNamefalse%
306             \@nameauth@FirstNametrue%
307         \else
308             \@nameauth@FullNametrue%
309             \@nameauth@FirstNamefalse%
310         \fi
311     \else
312         \unless\ifcsname#1\endcsname
313             \@nameauth@FullNametrue%
314             \@nameauth@FirstNamefalse%
315         \fi
316     \fi
317     \if@nameauth@FirstName
318         \@nameauth@FullNamefalse%
319     \fi
320     \ifx\FNN\@empty
321         \@nameauth@Hook{\SNN}%
322     \else
323         \if@nameauth@FullName

```



```

324         \if@nameauth@RevThis
325         \@nameauth@Hook{\FNN\Space\SNN}%
326     \else
327         \@nameauth@Hook{\SNN\Space\FNN}%
328     \fi
329 \else
330     \if@nameauth@FirstName
331         \if@nameauth@EastFN
332             \@nameauth@Hook{\FNN}%
333         \else
334             \@nameauth@Hook{\SNN}%
335         \fi
336     \else
337         \@nameauth@Hook{\SNN}%
338     \fi
339 \fi
340 \fi
341 \unless\ifcsname#1\endcsname
342     \unless\if@nameauth@InAKA\csgdef{#1}{}\fi
343 \fi
344 \@nameauth@FullNamefalse%
345 \@nameauth@FirstNamefalse%
346 \fi
347 }

```

**\@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.

```

348 \newcommand*\@nameauth@West[1]
349 {%
350     \if@nameauth@Lock
351         \unless\ifcsname#1\endcsname
352             \@nameauth@FirstFormattrue%
353         \fi
354     \if@nameauth@InAKA
355         \if@nameauth@AltAKA
356             \@nameauth@FullNamefalse%
357             \@nameauth@FirstNametrue%
358         \else
359             \@nameauth@FullNametrue%
360             \@nameauth@FirstNamefalse%
361         \fi
362     \else
363         \unless\ifcsname#1\endcsname
364             \@nameauth@FullNametrue%
365             \@nameauth@FirstNamefalse%
366         \fi
367     \fi
368     \if@nameauth@FirstName
369         \@nameauth@FullNamefalse%
370     \fi
371     \if@nameauth@FullName
372         \if@nameauth@RevThis
373             \@nameauth@Hook{\SNN\SpaceX\FNN}%
374         \else
375             \if@nameauth@RevThisComma

```

```

376         \edef\RevSpace{,\SpaceX}%
377         \@nameauth@Hook{\SNN\RevSpace\FNN}%
378     \else
379         \@nameauth@Hook{\FNN\SpaceX\SNN}%
380     \fi
381 \fi
382 \else
383     \if@nameauth@FirstName
384         \@nameauth@Hook{\FNN}%
385     \else
386         \@nameauth@Hook{\rootb}%
387     \fi
388 \fi
389 \unless\ifcsname#1\endcsname
390     \unless\if@nameauth@InAKA\csgdef{#1}{}\fi
391 \fi
392 \@nameauth@FullNamefalse%
393 \@nameauth@FirstNamefalse%
394 \fi
395 }

```

## 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.

```

396 \newcommand*\@nameauth@Hook[1]
397 {%
398     \if@nameauth@Lock
399         \@nameauth@InHooktrue%
400         \protected@edef\test{#1}%
401         \expandafter\@nameauth@TestDot\expandafter{\test}%
402         \if@nameauth@InAKA
403             \if@nameauth@AlwaysFormat
404                 \@nameauth@FirstFormattrue%
405             \else
406                 \unless\if@nameauth@AKAFormat
407                     \@nameauth@FirstFormatfalse\fi
408             \fi
409             \if@nameauth@MainFormat
410                 \if@nameauth@FirstFormat
411                     \bgroup\NamesFormat{#1}\egroup%
412                 \else
413                     \bgroup\MainNameHook{#1}\egroup%
414                 \fi
415             \else
416                 \if@nameauth@FirstFormat
417                     \bgroup\FrontNamesFormat{#1}\egroup%
418                 \else
419                     \bgroup\FrontNameHook{#1}\egroup%
420                 \fi
421             \fi
422         \else
423             \if@nameauth@AlwaysFormat
424                 \@nameauth@FirstFormattrue%
425             \fi

```

```

426     \if@nameauth@MainFormat
427     \if@nameauth@FirstFormat
428         \bgroup\NamesFormat{#1}\egroup%
429     \else
430         \bgroup\MainNameHook{#1}\egroup%
431     \fi
432 \else
433     \if@nameauth@FirstFormat
434         \bgroup\FrontNamesFormat{#1}\egroup%
435     \else
436         \bgroup\FrontNameHook{#1}\egroup%
437     \fi
438 \fi
439 \fi
440 \@nameauth@FirstFormatfalse%
441 \@nameauth@InHookfalse%
442 \fi
443 }

```

## Indexing Internals

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

```

444 \newcommand*\@nameauth@Index[2]
445 {%
446     \def\cseq{#1}%
447     \let\ex\expandafter%
448     \ifcsname\cseq!TAG\endcsname
449         \protected@edef\Tag{\csname#1!TAG\endcsname}%
450         \ex\def\ex\ShortTag\ex{\ex\@nameauth@TagRoot\ex{\Tag}}%
451     \fi
452     \if@nameauth@DoIndex
453         \ifcsname\cseq!TAG\endcsname
454             \ifcsname\cseq!PRE\endcsname
455                 \if@nameauth@Xref%
456                     \index%
457                     {\csname\cseq!PRE\endcsname#2\ShortTag}%
458                 \else
459                     \index%
460                     {\csname\cseq!PRE\endcsname#2\csname\cseq!TAG\endcsname}%
461                 \fi
462             \else
463                 \if@nameauth@Xref
464                     \index{#2\ShortTag}%
465                 \else
466                     \index{#2\csname\cseq!TAG\endcsname}%
467                 \fi
468             \fi
469         \else
470             \ifcsname\cseq!PRE\endcsname
471                 \index{\csname\cseq!PRE\endcsname#2}%
472             \else
473                 \index{#2}%
474             \fi
475         \fi
476     \fi
477 }

```

`\@nameauth@Actual` This sets the “actual” character used by `nameauth` for index sorting.  
478 `\newcommand*\@nameauth@Actual{@}`

### 3.5 User Interface Macros

#### Syntactic Formatting — Capitalization

`\CapThis` Tells the root capping macro to cap the first character. This excludes `\CapName`.  
479 `\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.  
480 `\newcommand*\AccentCapThis%`  
481 `{\@nameauth@Accenttrue\@nameauth@DoCapstrue}`

`\CapName` Capitalize entire required name. `\CapThis` overrides this.  
482 `\newcommand*\CapName{\@nameauth@AllThistrue}`

`\AllCapsInactive` Turn off global surname capitalization. `\CapThis` overrides this.  
483 `\newcommand*\AllCapsInactive{\@nameauth@AllCapsfalse}`

`\AllCapsActive` Turn on global surname capitalization. `\CapThis` overrides this.  
484 `\newcommand*\AllCapsActive{\@nameauth@AllCapstrue}`

#### Syntactic Formatting — Reversing

`\RevName` Reverse name order.  
485 `\newcommand*\RevName{\@nameauth@RevThistrue}`

`\ReverseInactive` Turn off global name reversing.  
486 `\newcommand*\ReverseInactive{\@nameauth@RevAllfalse}`

`\ReverseActive` Turn on global name reversing.  
487 `\newcommand*\ReverseActive{\@nameauth@RevAlltrue}`

`\ForceFN` Force the printing of an Eastern forename in the text, but only when using the “short name” macro `\FName` and the S-modifier.  
488 `\newcommand*\ForceFN{\@nameauth@EastFNtrue}`

#### Syntactic Formatting — Reversing with Commas

`\RevComma` Last name, comma, first name.  
489 `\newcommand*\RevComma%`  
490 `{\@nameauth@RevThisCommatrue}`

`\ReverseCommaInactive` Turn off global “last-name-comma-first.”  
491 `\newcommand*\ReverseCommaInactive%`  
492 `{\@nameauth@RevAllCommafalse}`

`\ReverseCommaActive` Turn on global “last-name-comma-first.”  
493 `\newcommand*\ReverseCommaActive%`  
494 `{\@nameauth@RevAllCommatrue}`

## Alternate Syntactic Formatting

`\AltFormatActive` Turn on alternate formatting.

```
495 \newcommand*\AltFormatActive{%
496   \global\@nameauth@AltFormattrue%
497   \global\@nameauth@DoAlttrue%
498 }
```

`\AltFormatActive*` Turn on alternate formatting.

```
499 \WithSuffix{\newcommand*}\AltFormatActive*{%
500   \global\@nameauth@AltFormattrue%
501   \global\@nameauth@DoAltfalse%
502 }
```

`\AltFormatInactive` Turn off alternate formatting.

```
503 \newcommand*\AltFormatInactive{%
504   \global\@nameauth@AltFormatfalse%
505   \global\@nameauth@DoAltfalse%
506 }
```

`\AltOn` Locally turn on alternate formatting.

```
507 \newcommand*\AltOn{%
508   \if@nameauth@InHook
509     \if@nameauth@AltFormat\@nameauth@DoAlttrue\fi
510   \fi
511 }
```

`\AltOff` Locally turn off alternate formatting.

```
512 \newcommand*\AltOff{%
513   \if@nameauth@InHook
514     \if@nameauth@AltFormat\@nameauth@DoAltfalse\fi
515   \fi
516 }
```

`\AltCaps` Alternate discretionary capping macro triggered by `\CapThis`.

```
517 \newcommand*\AltCaps[1]{%
518   \if@nameauth@InHook
519     \if@nameauth@DoCaps\uppercase{#1}\else#1\fi
520   \else#1%
521   \fi
522 }
```

`\textSC` Alternate formatting macro: small caps when active.

```
523 \newcommand*\textSC[1]{%
524   \if@nameauth@DoAlt\textsc{#1}\else#1\fi}
```

`\textUC` Alternate formatting macro: uppercase when active.

```
525 \newcommand*\textUC[1]{%
526   \if@nameauth@DoAlt\uppercase{#1}\else#1\fi}
```

`\textIT` Alternate formatting macro: italic when active.

```
527 \newcommand*\textIT[1]{%
528   \if@nameauth@DoAlt\textit{#1}\else#1\fi}
```

`\textBF` Alternate formatting macro: boldface when active.

```
529 \newcommand*\textBF[1]{%  
530   \if@nameauth@DoAlt\textbf{#1}\else#1\fi}
```

## Syntactic Formatting — Affixes

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

```
531 \newcommand*\ShowComma{\@nameauth@ShowCommtrue}
```

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

```
532 \newcommand*\NoComma{\@nameauth@NoCommtrue}
```

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

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

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

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

`\KeepName` Use non-breaking spaces between name syntactic forms.

```
535 \newcommand*\KeepName{\@nameauth@NBSPtrue\@nameauth@NBSPXtrue}
```

## Typographic Formatting — Main Versus Front Matter

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

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

`\NamesActive` Switch to the “formatted” species of names.

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

## Typographic Formatting — First / Subsequent Reference

`\ForgetThis` Have the naming engine `\@nameauth@Name` call `\ForgetName` internally.

```
538 \newcommand*\ForgetThis{\@nameauth@Forgettrue}
```

`\SubvertThis` Have the naming engine `\@nameauth@Name` call `\SubvertName` internally.

```
539 \newcommand*\SubvertThis{\@nameauth@Subverttrue}
```

`\ForceName` Set `\@nameauth@FirstFormat` to be true even for subsequent name uses. Works for one name only.

```
540 \newcommand*\ForceName{\@nameauth@FirstFormattrue}
```

## Name Occurrence Tweaks

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

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

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

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

## Index Operations

`\IndexInactive` Turn off global indexing of names.

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

|                           |  |
|---------------------------|--|
| <code>\SkipIndex</code>   | Turn off the next instance of indexing in <code>\Name</code> , <code>\FName</code> , and starred forms.<br>544 <code>\newcommand*\SkipIndex{\@nameauth@SkipIndextrue}</code>   |
| <code>\JustIndex</code>   | Makes the next call to <code>\Name</code> , <code>\FName</code> , and starred forms act like <code>\IndexName</code> . Overrides <code>\SkipIndex</code> .<br>545 <code>\newcommand*\JustIndex{\@nameauth@JustIndextrue}</code>                          |
| <code>\IndexActive</code> | Turn on global indexing of names.<br>546 <code>\newcommand*\IndexActive{\@nameauth@DoIndextrue}</code>   |
| <code>\IndexActual</code> | Change the “actual” character from the default.<br>547 <code>\newcommand*\IndexActual[1]</code><br>548 <code>{\global\renewcommand*\@nameauth@Actual{#1}}</code>   |
| <code>\SeeAlso</code>     | Change the type of cross-reference from a <i>see</i> reference to a <i>see also</i> reference. Works once per xref, unless one uses <code>\Include*</code> , in which case, take care!<br>549 <code>\newcommand*\SeeAlso{\@nameauth@SeeAlsottrue}</code> |

## 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.

```
550 \newcommand*\NameParser
551 {%
552   \if@nameauth@InHook
553     \let\SNN\rootb%
554     \ifx\arga\@empty
```

If the first optarg is empty, it is a non-Western name. The forename will be either the suffix or the final optarg.

```
555     \ifx\argc\@empty
556       \let\FNN\suffb%
557     \else
558       \let\FNN\argc%
559     \fi
560     \ifx\suffb\@empty
```

mononym

```
561     \ifx\FNN\@empty
562       \SNN%
563     \else
```

Eastern or ancient name, using the older syntax, with name reversing and forcing

```
564       \if@nameauth@FullName%
565       \if@nameauth@RevThis
566         \FNN\Space\SNN%
567       \else
568         \SNN\Space\FNN%
569       \fi
570     \else
571       \if@nameauth@FirstName
572       \if@nameauth@EastFN
573         \FNN%
574       \else
575         \SNN%
576       \fi
```

```

577         \else
578         \SNN%
579     \fi
580 \fi
581 \fi
582 \else

```

Eastern or ancient name, using the new syntax, with name reversing and forcing

```

583     \if@nameauth@FullName
584     \if@nameauth@RevThis
585     \FNN\Space\SNN%
586     \else
587     \SNN\Space\FNN%
588     \fi
589 \else
590     \if@nameauth@FirstName
591     \if@nameauth@EastFN
592     \FNN%
593     \else
594     \SNN%
595     \fi
596     \else
597     \SNN%
598     \fi
599 \fi
600 \fi
601 \else

```

Western name with name reversing and suffixes

```

602     \ifx\argc\@empty
603     \let\FNN\arga%
604 \else
605     \let\FNN\argc%
606 \fi
607 \unless\ifx\suffb\@empty
608     \protected@edef\SNN{\rootb\Space\suffb}%
609     \if@nameauth@ShortSNN\let\SNN\rootb\fi%
610 \fi
611 \if@nameauth@FullName
612     \if@nameauth@RevThis
613     \SNN\SpaceX\FNN%
614     \else
615     \if@nameauth@RevThisComma
616     \SNN\RevSpace\FNN%
617     \else
618     \FNN\SpaceX\SNN%
619     \fi
620 \fi
621 \else
622     \if@nameauth@FirstName
623     \FNN%
624     \else
625     \protected@edef\SNN{\rootb}%
626     \SNN%
627     \fi
628 \fi
629 \fi

```



```
630 \fi
631 }
```

## Traditional Naming Interface

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

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

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

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

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

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

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

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

## Index Operations

`\IndexName` This creates an index entry with page references. It issues warnings if the `verbose` option is selected. It prints nothing. First we make copies of the arguments.

```
638 \newcommandx*\IndexName[3][1=\@empty, 3=\@empty]
639 {%
640 \protected@edef\arga{\trim@spaces{#1}}%
641 \protected@edef\rootb{\@nameauth@Root{#2}}%
642 \protected@edef\suffb{\@nameauth@Suffix{#2}}%
643 \protected@edef\argc{\trim@spaces{#3}}%
644 \def\csb{\@nameauth@Clean{#2}}%
645 \def\csbc{\@nameauth@Clean{#2,#3}}%
646 \def\csab{\@nameauth@Clean{#1!#2}}%
```

Test for malformed input.

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

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`. If the first optarg is empty, it is a non-Western name.

```
648 \ifx\arga\@empty
649 \ifx\argc\@empty
650 \ifcsname\csb!PN\endcsname
651 \if@nameauth@Verbose
652 \PackageWarning{nameauth}%
653 {macro \IndexName: XRef: #2 exists}%
654 \fi
655 \else
656 \ifx\suffb\@empty
```

mononym or Eastern / ancient name, new syntax

```
657 \@nameauth@Index{\csb}{\rootb}%
658 \else
659 \@nameauth@Index{\csb}{\rootb\space\suffb}%
660 \fi
661 \fi
```

```

662 \else
663 \ifx\suffb\@empty
664 \ifcsname\csbc!PN\endcsname
665 \if@nameauth@Verbose
666 \PackageWarning{nameauth}%
667 {macro \IndexName: XRef: #2 #3 exists}%
668 \fi
669 \else
Eastern or ancient name, older syntax
670 \@nameauth@Index{\csbc}{\rootb\space\argc}%
671 \fi
672 \else
673 \ifcsname\csb!PN\endcsname
674 \if@nameauth@Verbose
675 \PackageWarning{nameauth}%
676 {macro \IndexName: XRef: #2 exists}%
677 \fi
678 \else
Eastern or ancient name, new syntax, alternate name ignored
679 \@nameauth@Index{\csb}{\rootb\space\suffb}%
680 \fi
681 \fi
682 \fi
683 \else
684 \ifcsname\csab!PN\endcsname
685 \if@nameauth@Verbose
686 \PackageWarning{nameauth}%
687 {macro \IndexName: XRef: #1 #2 exists}%
688 \fi
689 \else
Western name, without and with affix
690 \ifx\suffb\@empty
691 \@nameauth@Index{\csab}%
692 {\rootb,\space\arga}%
693 \else
694 \@nameauth@Index{\csab}%
695 {\rootb,\space\arga,\space\suffb}%
696 \fi
697 \fi
698 \fi
699 }

```

**\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.

```

700 \newcommandx*\IndexRef[4][1=\@empty, 3=\@empty]
701 {%
702 \protected@edef\arga{\trim@spaces{#1}}%
703 \protected@edef\rootb{\@nameauth@Root{#2}}%
704 \protected@edef\suffb{\@nameauth@Suffix{#2}}%
705 \protected@edef\argc{\trim@spaces{#3}}%
706 \protected@edef\target{#4}%
707 \def\csb{\@nameauth@Clean{#2}}%
708 \def\csbc{\@nameauth@Clean{#2,#3}}%
709 \def\csab{\@nameauth@Clean{#1!#2}}%

```

```
710 \let\ex\expandafter%
```

Test for malformed input.

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

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.

```
713 \ifx\arga\@empty
714 \ifx\argc\@empty
715 \ifcsname\csb!PN\endcsname
716 \if@nameauth@Verbose
717 \PackageWarning{nameauth}%
718 {macro \IndexRef: XRef: #2 exists}%
719 \fi
720 \else
721 \ifx\suffb\@empty
```

mononym or Eastern / ancient name, new syntax

```
722 \if@nameauth@SeeAlso
723 \@nameauth@Index{\csb}{\rootb|seealso{\target}}}%
724 \else
725 \@nameauth@Index{\csb}{\rootb|see{\target}}}%
726 \fi
727 \else
728 \if@nameauth@SeeAlso
729 \@nameauth@Index{\csb}{\rootb\space\suffb|seealso{\target}}}%
730 \else
731 \@nameauth@Index{\csb}{\rootb\space\suffb|see{\target}}}%
732 \fi
733 \fi
734 \csgdef{\csb!PN}{}%
735 \fi
736 \else
737 \ifx\suffb\@empty
738 \ifcsname\csbc!PN\endcsname
739 \if@nameauth@Verbose
740 \PackageWarning{nameauth}%
741 {macro \IndexRef: XRef: #2 #3 exists}%
742 \fi
743 \else
```

Eastern or ancient name, older syntax

```
744 \if@nameauth@SeeAlso
745 \@nameauth@Index{\csbc}%
746 {\rootb\space\argc|seealso{\target}}}%
747 \else
748 \@nameauth@Index{\csbc}%
749 {\rootb\space\argc|see{\target}}}%
750 \fi
751 \csgdef{\csbc!PN}{}%
752 \fi
753 \else
754 \ifcsname\csb!PN\endcsname
755 \if@nameauth@Verbose
756 \PackageWarning{nameauth}%
757 {macro \IndexRef: XRef: #2 exists}%
```

```

758         \fi
759     \else
Eastern or ancient name, new syntax, alternate name ignored
760         \if@nameauth@SeeAlso
761             \@nameauth@Index{\csb}%
762             {\rootb\space\suffb|seealso{\target}}}%
763         \else
764             \@nameauth@Index{\csb}%
765             {\rootb\space\suffb|see{\target}}}%
766         \fi
767     \csgdef{\csb!PN}{}%
768 \fi
769 \fi
770 \fi
771 \else
772     \ifcsname\csab!PN\endcsname
773     \if@nameauth@Verbose
774         \PackageWarning{nameauth}%
775         {macro \IndexRef: XRef: #1 #2 exists}%
776     \fi
777 \else
Western name, without and with affix
778     \ifx\suffb\@empty
779         \if@nameauth@SeeAlso
780             \@nameauth@Index{\csab}%
781             {\rootb,\space\arga|seealso{\target}}}%
782         \else
783             \@nameauth@Index{\csab}%
784             {\rootb,\space\arga|see{\target}}}%
785         \fi
786     \else
787         \if@nameauth@SeeAlso
788             \@nameauth@Index{\csab}%
789             {\rootb,\space\arga,\space\suffb|seealso{\target}}}%
790         \else
791             \@nameauth@Index{\csab}%
792             {\rootb,\space\arga,\space\suffb|see{\target}}}%
793         \fi
794     \fi
795     \csgdef{\csab!PN}{}%
796 \fi
797 \fi
798 \@nameauth@SeeAlsofalse%
799 \@nameauth@Xreffalse%
800 }

```

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

```

801 \newcommandx*\ExcludeName[3][1=\@empty, 3=\@empty]
802 {%
803     \protected@edef\arga{\trim@spaces{#1}}%
804     \protected@edef\argc{\trim@spaces{#3}}%
805     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
806     \def\csb{\@nameauth@Clean{#2}}%
807     \def\csbc{\@nameauth@Clean{#2,#3}}%
808     \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

809 \@nameauth@Error{#2}{macro \string\ExcludeName}%
810 \ifx\arga\@empty
811   \ifx\argc\@empty
812     \if@nameauth@Verbose
813       \ifcsname\csb!MN\endcsname
814         \PackageWarning{nameauth}%
815           {macro \ExcludeName: Reference: #2 exists}%
816       \fi
817     \ifcsname\csb!NF\endcsname
818       \PackageWarning{nameauth}%
819         {macro \ExcludeName: Reference: #2 exists}%
820     \fi
821   \fi
822   \ifcsname\csb!PN\endcsname
823     \if@nameauth@Verbose
824       \PackageWarning{nameauth}%
825         {macro \ExcludeName: Xref: #2 exists}%
826     \fi
827   \else
828     \csgdef{\csb!PN}{!}%
829   \fi
830 \else
831   \ifx\suffb\@empty
832     \if@nameauth@Verbose
833       \ifcsname\csbc!MN\endcsname
834         \PackageWarning{nameauth}%
835           {macro \ExcludeName: Reference: #2 #3 exists}%
836       \fi
837     \ifcsname\csbc!NF\endcsname
838       \PackageWarning{nameauth}%
839         {macro \ExcludeName: Reference: #2 #3 exists}%
840     \fi
841   \fi
842   \csgdef{\csbc!PN}{!}%
843   \ifcsname\csbc!PN\endcsname
844     \if@nameauth@Verbose
845       \PackageWarning{nameauth}%
846         {macro \ExcludeName: Xref: #2 exists}%
847     \fi
848   \else
849     \csgdef{\csbc!PN}{!}%
850   \fi
851 \else
852   \if@nameauth@Verbose
853     \ifcsname\csb!MN\endcsname
854       \PackageWarning{nameauth}%
855         {macro \ExcludeName: Reference: #2 exists}%
856     \fi
857   \ifcsname\csb!NF\endcsname
858     \PackageWarning{nameauth}%
859       {macro \ExcludeName: Reference: #2 exists}%
860   \fi
861 \fi
862 \ifcsname\csb!PN\endcsname
863   \if@nameauth@Verbose

```

```

864         \PackageWarning{nameauth}%
865         {macro \ExcludeName: Xref: #2 exists}%
866     \fi
867     \else
868         \csgdef{\csb!PN}{!}%
869     \fi
870 \fi
871 \fi
872 \else
873     \if@nameauth@Verbose
874         \ifcsname\csab!MN\endcsname
875             \PackageWarning{nameauth}%
876             {macro \ExcludeName: Reference: #1 #2 exists}%
877         \fi
878         \ifcsname\csab!NF\endcsname
879             \PackageWarning{nameauth}%
880             {macro \ExcludeName: Reference: #1 #2 exists}%
881         \fi
882     \fi
883     \ifcsname\csab!PN\endcsname
884         \if@nameauth@Verbose
885             \PackageWarning{nameauth}%
886             {macro \ExcludeName: Xref: #2 exists}%
887         \fi
888     \else
889         \csgdef{\csab!PN}{!}%
890     \fi
891 \fi
892 }

```

`\IncludeName` This macro allows a name to be indexed if it is not a cross-reference.

```

893 \newcommandx*\IncludeName[3][1=\@empty, 3=\@empty]
894 {%
895     \protected@edef\arga{\trim@spaces{#1}}%
896     \protected@edef\argc{\trim@spaces{#3}}%
897     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
898     \def\csb{\@nameauth@Clean{#2}}%
899     \def\csbc{\@nameauth@Clean{#2,#3}}%
900     \def\csab{\@nameauth@Clean{#1!#2}}%

```

Below we parse the name arguments and undefine an “excluded” name.

```

901 \@nameauth@Error{#2}{macro \string\IncludeName}%
902 \ifx\arga\@empty
903     \ifx\argc\@empty
904         \ifcsname\csb!PN\endcsname
905             \edef\testex{\csname\csb!PN\endcsname}%
906             \unless\ifx\testex\@empty\global\csundef{\csb!PN}\fi
907         \fi
908     \else
909         \ifx\suffb\@empty
910             \ifcsname\csbc!PN\endcsname
911                 \edef\testex{\csname\csbc!PN\endcsname}%
912                 \unless\ifx\testex\@empty\global\csundef{\csbc!PN}\fi
913             \fi
914         \else
915             \ifcsname\csb!PN\endcsname

```

```

916         \edef\testex{\csname\csb!PN\endcsname}%
917         \unless\ifx\testex\@empty\global\csundef{\csb!PN}\fi
918     \fi
919 \fi
920 \fi
921 \else
922     \ifcsname\csab!PN\endcsname
923         \edef\testex{\csname\csab!PN\endcsname}%
924         \unless\ifx\testex\@empty\global\csundef{\csab!PN}\fi
925     \fi
926 \fi
927 }

```

**\IncludeName\*** This macro allows any name to be indexed, undefining cross-references.

```

928 \WithSuffix{\newcommandx*}\IncludeName*[3][1=\@empty, 3=\@empty]
929 {%
930     \protected@edef\arga{\trim@spaces{#1}}%
931     \protected@edef\argc{\trim@spaces{#3}}%
932     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
933     \def\csb{\@nameauth@Clean{#2}}%
934     \def\csbc{\@nameauth@Clean{#2,#3}}%
935     \def\csab{\@nameauth@Clean{#1!#2}}%

```

Below we parse the name arguments and undefine an xref control sequence.

```

936     \@nameauth@Error{#2}{macro \string\IncludeName*}%
937     \ifx\arga\@empty
938         \ifx\argc\@empty
939             \global\csundef{\csb!PN}%
940         \else
941             \ifx\suffb\@empty
942                 \global\csundef{\csbc!PN}%
943             \else
944                 \global\csundef{\csb!PN}%
945             \fi
946         \fi
947     \else
948         \global\csundef{\csab!PN}%
949     \fi
950 }

```

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

```

951 \newcommandx*\PretagName[4][1=\@empty, 3=\@empty]
952 {%
953     \protected@edef\arga{\trim@spaces{#1}}%
954     \protected@edef\argc{\trim@spaces{#3}}%
955     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
956     \def\csb{\@nameauth@Clean{#2}}%
957     \def\csbc{\@nameauth@Clean{#2,#3}}%
958     \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

959     \@nameauth@Error{#2}{macro \string\PretagName}%
960     \ifx\arga\@empty
961         \ifx\argc\@empty
962             \ifcsname\csb!PN\endcsname

```

```

963         \if@nameauth@Verbose
964         \PackageWarning{nameauth}%
965         {macro \PretagName: tagging xref: #2}%
966         \fi
967     \fi
968     \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
969 \else
970     \ifx\suffb\@empty
971         \ifcsname\csbc!PN\endcsname
972         \if@nameauth@Verbose
973         \PackageWarning{nameauth}%
974         {macro \PretagName: tagging xref: #2 #3}%
975         \fi
976     \fi
977     \if@nameauth@Pretag\csgdef{\csbc!PRE}{#4\@nameauth@Actual}\fi
978 \else
979     \ifcsname\csb!PN\endcsname
980     \if@nameauth@Verbose
981     \PackageWarning{nameauth}%
982     {macro \PretagName: tagging xref: #2}%
983     \fi
984 \fi
985     \if@nameauth@Pretag\csgdef{\csb!PRE}{#4\@nameauth@Actual}\fi
986 \fi
987 \fi
988 \else
989     \ifcsname\csab!PN\endcsname
990     \if@nameauth@Verbose
991     \PackageWarning{nameauth}%
992     {macro \PretagName: tagging xref: #1 #2}%
993     \fi
994 \fi
995     \if@nameauth@Pretag\csgdef{\csab!PRE}{#4\@nameauth@Actual}\fi
996 \fi
997 }

```

`\TagName` This creates an index entry tag for a name that is not used as a cross-reference.

```

998 \newcommandx*\TagName[4][1=\@empty, 3=\@empty]
999 {%
1000     \protected@edef\arga{\trim@spaces{#1}}%
1001     \protected@edef\argc{\trim@spaces{#3}}%
1002     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1003     \def\csb{\@nameauth@Clean{#2}}%
1004     \def\csbc{\@nameauth@Clean{#2,#3}}%
1005     \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1006     \@nameauth@Error{#2}{macro \string\TagName}%
1007     \ifx\arga\@empty
1008     \ifx\argc\@empty
1009     \ifcsname\csb!PN\endcsname
1010     \if@nameauth@Verbose
1011     \PackageWarning{nameauth}%
1012     {macro \TagName: not tagging xref: #2}%
1013     \fi

```



```

1014     \else
1015     \csgdef{\csb!TAG}{#4}%
1016     \fi
1017 \else
1018     \ifx\suffb\@empty
1019     \ifcsname\csbc!PN\endcsname
1020     \if@nameauth@Verbose
1021     \PackageWarning{nameauth}%
1022     {macro \TagName: not tagging xref: #2 #3}%
1023     \fi
1024     \else
1025     \csgdef{\csbc!TAG}{#4}%
1026     \fi
1027 \else
1028     \ifcsname\csb!PN\endcsname
1029     \if@nameauth@Verbose
1030     \PackageWarning{nameauth}%
1031     {macro \TagName: not tagging xref: #2}%
1032     \fi
1033     \else
1034     \csgdef{\csb!TAG}{#4}%
1035     \fi
1036 \fi
1037 \fi
1038 \else
1039     \ifcsname\csab!PN\endcsname
1040     \if@nameauth@Verbose
1041     \PackageWarning{nameauth}%
1042     {macro \TagName: not tagging xref: #1 #2}%
1043     \fi
1044     \else
1045     \csgdef{\csab!TAG}{#4}%
1046     \fi
1047 \fi
1048 }

```

`\UntagName` This deletes an index tag.

```

1049 \newcommandx*\UntagName[3][1=\@empty, 3=\@empty]
1050 {%
1051     \protected@edef\arga{\trim@spaces{#1}}%
1052     \protected@edef\argc{\trim@spaces{#3}}%
1053     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1054     \def\csb{\@nameauth@Clean{#2}}%
1055     \def\csbc{\@nameauth@Clean{#2,#3}}%
1056     \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1057     \@nameauth@Error{#2}{macro \string\UntagName}%
1058     \ifx\arga\@empty
1059     \ifx\argc\@empty
1060     \global\csundef{\csb!TAG}%
1061     \else
1062     \ifx\suffb\@empty
1063     \global\csundef{\csbc!TAG}%
1064     \else
1065     \global\csundef{\csb!TAG}%

```

```

1066     \fi
1067 \fi
1068 \else
1069     \global\csundef{\csab!TAG}%
1070 \fi
1071 }

```

## Name Info Data Set: “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.

```

1072 \newcommandx\NameAddInfo[4][1=\@empty, 3=\@empty]
1073 {%
1074     \protected@edef\arga{\trim@spaces{#1}}%
1075     \protected@edef\argc{\trim@spaces{#3}}%
1076     \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1077     \def\csb{\@nameauth@Clean{#2}}%
1078     \def\csbc{\@nameauth@Clean{#2,#3}}%
1079     \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1080     \@nameauth@Error{#2}{macro \string\NameAddInfo}%
1081     \ifx\arga\@empty
1082         \ifx\argc\@empty
1083             \csgdef{\csb!DB}{#4}%
1084         \else
1085             \ifx\Suff\@empty
1086                 \csgdef{\csbc!DB}{#4}%
1087             \else
1088                 \csgdef{\csb!DB}{#4}%
1089             \fi
1090         \fi
1091     \else
1092         \csgdef{\csab!DB}{#4}%
1093     \fi
1094 }

```

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

```

1095 \newcommandx*\NameQueryInfo[3][1=\@empty, 3=\@empty]
1096 {%
1097     \protected@edef\arga{\trim@spaces{#1}}%
1098     \protected@edef\argc{\trim@spaces{#3}}%
1099     \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1100     \def\csb{\@nameauth@Clean{#2}}%
1101     \def\csbc{\@nameauth@Clean{#2,#3}}%
1102     \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1103     \@nameauth@Error{#2}{macro \string\NameQueryInfo}%
1104     \ifx\arga\@empty
1105         \ifx\argc\@empty
1106             \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
1107         \else
1108             \ifx\Suff\@empty
1109                 \ifcsname\csbc!DB\endcsname\csname\csbc!DB\endcsname\fi
1110             \else

```

```

1111      \ifcsname\csb!DB\endcsname\csname\csb!DB\endcsname\fi
1112      \fi
1113      \fi
1114      \else
1115      \ifcsname\csab!DB\endcsname\csname\csab!DB\endcsname\fi
1116      \fi
1117 }

```

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

```

1118 \newcommandx*\NameClearInfo[3][1=\@empty, 3=\@empty]
1119 {%
1120   \protected@edef\arga{\trim@spaces{#1}}%
1121   \protected@edef\argc{\trim@spaces{#3}}%
1122   \protected@edef\Suff{\@nameauth@Suffix{#2}}%
1123   \def\csb{\@nameauth@Clean{#2}}%
1124   \def\csbc{\@nameauth@Clean{#2,#3}}%
1125   \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1126   \@nameauth@Error{#2}{macro \string\NameClearInfo}%
1127   \ifx\arga\@empty
1128     \ifx\argc\@empty
1129       \global\csundef{\csb!DB}%
1130     \else
1131       \ifx\Suff\@empty
1132         \global\csundef{\csbc!DB}%
1133       \else
1134         \global\csundef{\csb!DB}%
1135       \fi
1136     \fi
1137   \else
1138     \global\csundef{\csab!DB}%
1139   \fi
1140 }

```

## Name Decisions

**\IfMainName** This macro expands one path if a main matter name exists, or else the other.

```

1141 \newcommandx\IfMainName[5][1=\@empty, 3=\@empty]
1142 {%
1143   \protected@edef\arga{\trim@spaces{#1}}%
1144   \protected@edef\argc{\trim@spaces{#3}}%
1145   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1146   \def\csb{\@nameauth@Clean{#2}}%
1147   \def\csbc{\@nameauth@Clean{#2,#3}}%
1148   \def\csab{\@nameauth@Clean{#1!#2}}%

```

Below we parse the name arguments and choose the path.

```

1149   \@nameauth@Error{#2}{macro \string\IfMainName}%
1150   \ifx\arga\@empty
1151     \ifx\argc\@empty
1152       \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
1153     \else
1154       \ifx\suffb\@empty
1155         \ifcsname\csbc!MN\endcsname{#4}\else{#5}\fi
1156       \else

```

```

1157         \ifcsname\csb!MN\endcsname{#4}\else{#5}\fi
1158     \fi
1159 \fi
1160 \else
1161     \ifcsname\csab!MN\endcsname{#4}\else{#5}\fi
1162 \fi
1163 }

```

**\IfFrontName** This macro expands one path if a front matter name exists, or else the other.

```

1164 \newcommandx\IfFrontName[5][1=\@empty, 3=\@empty]
1165 {%
1166     \protected@edef\arga{\trim@spaces{#1}}%
1167     \protected@edef\argc{\trim@spaces{#3}}%
1168     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1169     \def\csb{\@nameauth@Clean{#2}}%
1170     \def\csbc{\@nameauth@Clean{#2,#3}}%
1171     \def\csab{\@nameauth@Clean{#1!#2}}%

```

Below we parse the name arguments and choose the path.

```

1172     \@nameauth@Error{#2}{macro \string\IfFrontName}%
1173     \ifx\arga\@empty
1174         \ifx\argc\@empty
1175             \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
1176         \else
1177             \ifx\suffb\@empty
1178                 \ifcsname\csbc!NF\endcsname{#4}\else{#5}\fi
1179             \else
1180                 \ifcsname\csb!NF\endcsname{#4}\else{#5}\fi
1181             \fi
1182         \fi
1183     \else
1184         \ifcsname\csab!NF\endcsname{#4}\else{#5}\fi
1185     \fi
1186 }

```

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

```

1187 \newcommandx\IfAKA[6][1=\@empty, 3=\@empty]
1188 {%
1189     \protected@edef\arga{\trim@spaces{#1}}%
1190     \protected@edef\argc{\trim@spaces{#3}}%
1191     \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1192     \def\csb{\@nameauth@Clean{#2}}%
1193     \def\csbc{\@nameauth@Clean{#2,#3}}%
1194     \def\csab{\@nameauth@Clean{#1!#2}}%

```

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

```

1195     \@nameauth@Error{#2}{macro \string\IfAKA}%
1196     \ifx\arga\@empty
1197         \ifx\argc\@empty
1198             \ifcsname\csb!PN\endcsname
1199                 \edef\testex{\csname\csb!PN\endcsname}%
1200                 \ifx\testex\@empty{#4}\else{#6}\fi
1201             \else{#5}\fi
1202         \else
1203             \ifx\suffb\@empty

```

```

1204     \ifcsname\csbc!PN\endcsname
1205     \edef\testex{\csname\csbc!PN\endcsname}%
1206     \ifx\testex\@empty{#4}\else{#6}\fi
1207     \else{#5}\fi
1208   \else
1209     \ifcsname\csb!PN\endcsname
1210     \edef\testex{\csname\csb!PN\endcsname}%
1211     \ifx\testex\@empty{#4}\else{#6}\fi
1212     \else{#5}\fi
1213   \fi
1214 \fi
1215 \else
1216   \ifcsname\csab!PN\endcsname
1217   \edef\testex{\csname\csab!PN\endcsname}%
1218   \ifx\testex\@empty{#4}\else{#6}\fi
1219   \else{#5}\fi
1220 \fi
1221 }

```

## Changing Name Decisions

**\ForgetName** This undefines a control sequence to force a “first use.”

```

1222 \newcommandx*\ForgetName[3][1=\@empty, 3=\@empty]
1223 {%
1224   \protected@edef\arga{\trim@spaces{#1}}%
1225   \protected@edef\argc{\trim@spaces{#3}}%
1226   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1227   \def\csb{\@nameauth@Clean{#2}}%
1228   \def\csbc{\@nameauth@Clean{#2,#3}}%
1229   \def\csab{\@nameauth@Clean{#1!#2}}%
1230   \@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`).

```

1231   \ifx\arga\@empty
1232     \ifx\argc\@empty
1233       \if@nameauth@LocalNames
1234         \if@nameauth@MainFormat
1235           \global\csundef{\csb!MN}%
1236         \else
1237           \global\csundef{\csb!NF}%
1238         \fi
1239       \else
1240         \global\csundef{\csb!MN}%
1241         \global\csundef{\csb!NF}%
1242       \fi
1243     \else
1244       \ifx\suffb\@empty
1245         \if@nameauth@LocalNames
1246           \if@nameauth@MainFormat
1247             \global\csundef{\csbc!MN}%
1248           \else
1249             \global\csundef{\csbc!NF}%
1250           \fi
1251         \else
1252           \global\csundef{\csbc!MN}%
1253           \global\csundef{\csbc!NF}%

```

```

1254     \fi
1255   \else
1256     \if@nameauth@LocalNames
1257       \if@nameauth@MainFormat
1258         \global\csundef{\csb!MN}%
1259       \else
1260         \global\csundef{\csb!NF}%
1261       \fi
1262     \else
1263       \global\csundef{\csb!MN}%
1264       \global\csundef{\csb!NF}%
1265     \fi
1266   \fi
1267 \fi
1268 \else
1269   \if@nameauth@LocalNames
1270     \if@nameauth@MainFormat
1271       \global\csundef{\csab!MN}%
1272     \else
1273       \global\csundef{\csab!NF}%
1274     \fi
1275   \else
1276     \global\csundef{\csab!MN}%
1277     \global\csundef{\csab!NF}%
1278   \fi
1279 \fi
1280 }

```

`\SubvertName` This defines a control sequence to force a “subsequent use.”

```

1281 \newcommandx*\SubvertName[3][1=\@empty, 3=\@empty]
1282 {%
1283   \protected@edef\arga{\trim@spaces{#1}}%
1284   \protected@edef\argc{\trim@spaces{#3}}%
1285   \protected@edef\suffb{\@nameauth@Suffix{#2}}%
1286   \def\csb{\@nameauth@Clean{#2}}%
1287   \def\csbc{\@nameauth@Clean{#2,#3}}%
1288   \def\csab{\@nameauth@Clean{#1!#2}}%

```

We make copies of the arguments to test them.

```

1289   \@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 we select the type of name with `@nameauth@MainFormat`.

```

1290   \ifx\arga\@empty
1291     \ifx\argc\@empty
1292       \if@nameauth@LocalNames
1293         \if@nameauth@MainFormat
1294           \csgdef{\csb!MN}{}%
1295         \else
1296           \csgdef{\csb!NF}{}%
1297         \fi
1298       \else
1299         \csgdef{\csb!MN}{}%
1300         \csgdef{\csb!NF}{}%
1301       \fi

```

```

1302 \else
1303 \ifx\suffb\@empty
1304 \if@nameauth@LocalNames
1305 \if@nameauth@MainFormat
1306 \csgdef{\csbc!MN}{}%
1307 \else
1308 \csgdef{\csbc!NF}{}%
1309 \fi
1310 \else
1311 \csgdef{\csbc!MN}{}%
1312 \csgdef{\csbc!NF}{}%
1313 \fi
1314 \else
1315 \if@nameauth@LocalNames
1316 \if@nameauth@MainFormat
1317 \csgdef{\csb!MN}{}%
1318 \else
1319 \csgdef{\csb!NF}{}%
1320 \fi
1321 \else
1322 \csgdef{\csb!MN}{}%
1323 \csgdef{\csb!NF}{}%
1324 \fi
1325 \fi
1326 \fi
1327 \else
1328 \if@nameauth@LocalNames
1329 \if@nameauth@MainFormat
1330 \csgdef{\csab!MN}{}%
1331 \else
1332 \csgdef{\csab!NF}{}%
1333 \fi
1334 \else
1335 \csgdef{\csab!MN}{}%
1336 \csgdef{\csab!NF}{}%
1337 \fi
1338 \fi
1339 }

```

## Alternate Names

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

```

1340 \newcommandx*\AKA[5][1=\@empty, 3=\@empty, 5=\@empty]
1341 {%

```

Prevent entering `\AKA` via itself or `\@nameauth@Name`. Prevent the index-only flag.

```

1342 \unless\if@nameauth@Lock
1343 \@nameauth@Locktrue%
1344 \@nameauth@JustIndexfalse%

```

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

```

1345 \@nameauth@InAKAtrue%
1346 \@nameauth@Error{#2}{\macro \string\AKA}%
1347 \@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.

```

1348 \leavevmode\hbox{%
1349 \protected@edef\argi{\trim@spaces{#1}}%
1350 \protected@edef\rooti{\@nameauth@Root{#2}}%
1351 \protected@edef\suffi{\@nameauth@Suffix{#2}}%
1352 \@nameauth@Parse[#3]{#4}[#5]{!PN}%

```

Create an index cross-reference based on the arguments.

```

1353 \unless\if@nameauth@SkipIndex
1354 \ifx\argi\@empty
1355 \ifx\suffi\@empty
1356 \IndexRef[#3]{#4}[#5]{\rooti}%
1357 \else
1358 \IndexRef[#3]{#4}[#5]{\rooti\space\suffi}%
1359 \fi
1360 \else
1361 \ifx\suffi\@empty
1362 \IndexRef[#3]{#4}[#5]{\rooti,\space\argi}%
1363 \else
1364 \IndexRef[#3]{#4}[#5]{\rooti,\space\argi,\space\suffi}%
1365 \fi
1366 \fi
1367 \fi

```

Reset all the “per name” Boolean values.

```

1368 \@nameauth@SkipIndexfalse%
1369 \@nameauth@Lockfalse%
1370 \@nameauth@InAKAfalse%
1371 \@nameauth@AltAKAfalse%
1372 \@nameauth@NBSPfalse%
1373 \@nameauth@NBSPXfalse%
1374 \@nameauth@DoCapsfalse%
1375 \@nameauth@Accentfalse%
1376 \@nameauth@AllThisfalse%
1377 \@nameauth@ShowCommafalse%
1378 \@nameauth@NoCommafalse%
1379 \@nameauth@RevThisfalse%
1380 \@nameauth@RevThisCommafalse%
1381 \@nameauth@ShortSNNfalse%
1382 \@nameauth@EastFNfalse%

```

Close the “locked” branch.

```

1383 \fi

```

Call the full stop detection.

```

1384 \if@nameauth@Punct\expandafter\@nameauth@CheckDot\fi
1385 }

```

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

```

1386 \WithSuffix{\newcommand*}\AKA*{\@nameauth@AltAKAtrue\AKA}

```

**\PName** \PName is a convenience macro that calls \NameauthName, then \AKA. It prevents the index-only feature from triggering.

```

1387 \newcommandx*\PName[5][1=\@empty,3=\@empty,5=\@empty]
1388 {%
1389 \@nameauth@JustIndexfalse%
1390 \if@nameauth@SkipIndex

```



```

1391 \NameauthName[#1]{#2}\space(\SkipIndex\AKA[#1]{#2}[#3]{#4}[#5])%
1392 \else
1393 \NameauthName[#1]{#2}\space(\AKA[#1]{#2}[#3]{#4}[#5])%
1394 \fi
1395 }

```

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

```

1396 \WithSuffix{\newcommand*}\PName*{\@nameauth@FullNametrue\PName}

```

## Simplified Interface

`nameauth` The `nameauth` environment creates macro shorthands. First we define a control sequence `\<` that takes four parameters, delimited by three ampersands and `>`.

```

1397 \newenvironment{nameauth}{%
1398 \begingroup%
1399 \let\ex\expandafter%
1400 \csdef{<###1&##2&##3&##4>{%
1401 \protected@edef\@arga{\trim@spaces{##1}}%
1402 \protected@edef\@testb{\trim@spaces{##2}}%
1403 \protected@edef\@testd{\trim@spaces{##4}}%
1404 \@nameauth@etoksb\expandafter{##2}%
1405 \@nameauth@etoksc\expandafter{##3}%
1406 \@nameauth@etoksd\expandafter{##4}%

```

The first argument must have some text to create a set of control sequences with it. The third argument is the required name field. Redefining a shorthand creates a warning.

```

1407 \ifx\@arga\@empty
1408 \PackageError{nameauth}%
1409 {environment nameauth: Control sequence missing}%
1410 \fi
1411 \@nameauth@Error{##3}{environment nameauth}%
1412 \ifcsname\@arga\endcsname
1413 \PackageWarning{nameauth}%
1414 {environment nameauth: Shorthand macro already exists}%
1415 \fi

```

Set up shorthands according to name form. We have to use `\expandafter`, not the  $\epsilon$ -TEX way, due to `\protected@edef` in the naming macros.

We begin with mononyms and non-Western names that use the new syntax. We use one `\expandafter` per token because we only have one argument to expand first.

```

1416 \ifx\@testd\@empty
1417 \ifx\@testb\@empty
1418 \ex\csgdef\ex{\ex\@arga\ex}\ex{\ex\NameauthName\ex}%
1419 \the\@nameauth@etoksc}%
1420 \ex\csgdef\ex{\ex L\ex\@arga\ex}\ex{%
1421 \ex\@nameauth@FullNametrue%
1422 \ex\NameauthLName\ex{\the\@nameauth@etoksc}}%
1423 \ex\csgdef\ex{\ex S\ex\@arga\ex}\ex{%
1424 \ex\@nameauth@FirstNametrue%
1425 \ex\NameauthFName\ex{\the\@nameauth@etoksc}}%
1426 \else

```

Next we have Western names with no alternate names. Here we have two arguments to expand in reverse order, so we need three, then one uses of `\expandafter` per token.

```

1427 \ex\ex\ex\csgdef\ex\ex\ex{\ex\ex\ex\@arga\ex\ex\ex}%
1428 \ex\ex\ex{\ex\ex\ex\NameauthName\ex\ex\ex[%
1429 \ex\the\ex\@nameauth@etoksb\ex]\ex{\the\@nameauth@etoksc}}%

```



## 4 Change History

|      |   |    |      |  |    |
|------|---|----|------|--|----|
| 0.7  | General: Initial release . . . . .              | 1  | 1.6  | nameauth: Environment added . .  | 97 |
| 0.75 | \ForgetName: New argument added                 | 93 | 1.9  | \ForgetName: Ensure global undef   | 93 |
|      | \IndexName: Current arguments .                 | 81 |      | \KeepAffix: Added . . . . .  | 78 |
| 0.85 | \@nameauth@Name: Comma suppression . . . . .    | 69 |      | \TagName: Fix cs collisions . . . .  | 88 |
|      | \AKA: Comma suppression . . . . .               | 95 |      | \UntagName: Ensure global undef, fix cs collisions . . . . .                   | 89 |
|      | \IndexName: Comma suppression                   | 81 |      | nameauth: Bugfix . . . . .   | 97 |
| 0.9  | \@nameauth@Suffix: Added . . . .                | 67 | 2.0  | \@nameauth@Actual: Added . . . .   | 76 |
|      | \@nameauth@TrimRoot: Expandable . . . . .       | 67 |      | \@nameauth@Index: New tagging  | 75 |
|      | \@nameauth@TrimSuffix: Added                    | 67 |      | \@nameauth@Name: Trim spaces; redesign tagging . . . . .                       | 69 |
|      | \AKA*: Added . . . . .                          | 96 |      | \@nameauth@TrimRoot: Trim spaces . . . . .                                     | 67 |
|      | \FName: Added . . . . .                         | 81 |      | \AKA: Trim spaces; fix tagging . .   | 95 |
|      | \SubvertName: Added . . . . .                   | 94 |      | \IndexActual: Added . . . . .  | 79 |
| 0.94 | \@nameauth@Hook: Particle caps .                | 74 |      | \IndexName: Trim spaces; redesign tagging . . . . .                            | 81 |
|      | \@nameauth@Index: Added . . . . .               | 75 |      | \PretagName: Added . . . . .   | 87 |
|      | \CapThis: Added . . . . .                       | 76 |      | \TagName: Redesign tagging . . . .   | 88 |
|      | \ExcludeName: Added . . . . .                   | 84 |      | \UntagName: Redesign tagging . .   | 89 |
|      | \IndexActive: Added . . . . .                   | 79 |      | General: Use dtxgen template; prevent malformed input in most macros . . . . . | 1  |
|      | \IndexInactive: Added . . . . .                 | 78 |      | nameauth: Redesigned argument handling . . . . .                               | 97 |
| 0.95 | \@nameauth@Hook: Works with microtype . . . . . | 74 | 2.1  | \@nameauth@Name: Isolate Unicode issues . . . . .                              | 69 |
| 1.2  | \TagName: Added . . . . .                       | 88 |      | \AKA: Fix Unicode issues . . . . .   | 95 |
|      | \UntagName: Added . . . . .                     | 89 |      | \AccentCapThis: Added . . . . .  | 76 |
| 1.26 | \AKA: Fix name suffixes . . . . .               | 95 | 2.11 | nameauth: Bugfix . . . . .   | 97 |
|      | \IndexName: Fix name suffix sorting . . . . .   | 81 | 2.2  | \NameauthFName: Added . . . . .  | 65 |
| 1.4  | \@nameauth@Root: More robust .                  | 66 |      | \NameauthName: Added . . . . .   | 65 |
|      | \ShowComma: Added . . . . .                     | 78 | 2.3  | \@nameauth@Name: Now internal .  | 69 |
| 1.5  | \@nameauth@Name: Reversing/caps                 | 69 |      | \AKA: Expand starred mode . . . .  | 95 |
|      | \@nameauth@TrimSuffix: Trim spaces . . . . .    | 67 |      | \ExcludeName: Make special xref type . . . . .                                 | 84 |
|      | \AKA: Reversing and caps . . . . .              | 95 |      | \FName: Interface macro . . . . .  | 81 |
|      | \AllCapsActive: Added . . . . .                 | 76 |      | \FName*: Interface macro . . . . .   | 81 |
|      | \AllCapsInactive: Added . . . . .               | 76 |      | \ForgetName: Global or local . .   | 93 |
|      | \CapName: Added . . . . .                       | 76 |      | \GlobalNames: Added . . . . .  | 78 |
|      | \RevComma: Added . . . . .                      | 76 |      | \IfAKA: Added . . . . .  | 92 |
|      | \RevName: Added . . . . .                       | 76 |      | \IfFrontName: Added . . . . .  | 92 |
|      | \ReverseActive: Added . . . . .                 | 76 |      | \IfMainName: Added . . . . .   | 91 |
|      | \ReverseCommaActive: Added . .                  | 76 |      | \LocalNames: Added . . . . .   | 78 |
|      | \ReverseCommaInactive: Added                    | 76 |      | \Name: Interface macro . . . . .   | 81 |
|      | \ReverseInactive: Added . . . .                 | 76 |      | \Name*: Interface macro . . . . .  | 81 |

|      |                                    |    |      |                                   |    |
|------|------------------------------------|----|------|-----------------------------------|----|
|      | \NameauthLName: Added .....        | 65 |      | \IfAKA: Redesigned .....          | 92 |
|      | \PName: Work directly with hooks   | 96 |      | \IncludeName: Added .....         | 86 |
|      | \SubvertName: Global or local ..   | 94 |      | \IncludeName*: Added .....        | 87 |
| 2.4  |                                    |    |      | \IndexName: Redesigned .....      | 81 |
|      | \@nameauth@Hook: Add hooks ...     | 74 |      | \IndexRef: Added .....            | 82 |
|      | \@nameauth@Name: Add token regs    |    |      | \NameParser: Added .....          | 79 |
|      | for hooks .....                    | 69 |      | \SeeAlso: Added .....             | 79 |
|      | \AKA: Fix formatting; add token    |    | 3.01 |                                   |    |
|      | regs .....                         | 95 |      | \@nameauth@Error: Fixed .....     | 68 |
|      | \FrontNameHook: Added .....        | 65 | 3.02 |                                   |    |
|      | \GlobalNames: Ensured to be        |    |      | \@nameauth@NonWest: Restrict      |    |
|      | global .....                       | 78 |      | \ForceFN .....                    | 72 |
|      | \IfAKA: New exclusion test .....   | 92 | 3.03 |                                   |    |
|      | \LocalNames: Ensured to be global  | 78 |      | \NameParser: First name only with |    |
|      | \MainNameHook: Added .....         | 65 |      | “short” macros .....              | 79 |
|      | \NameAddInfo: Added .....          | 90 | 3.1  |                                   |    |
|      | \NameClearInfo: Added .....        | 91 |      | \@nameauth@Cap: Added; old caps   |    |
|      | \NameQueryInfo: Added .....        | 90 |      | obsolete .....                    | 67 |
| 2.41 |                                    |    |      | \@nameauth@Cii: Added .....       | 67 |
|      | \@nameauth@Name: No local          |    |      | \@nameauth@Ciii: Added .....      | 68 |
|      | \newtoks .....                     | 69 |      | \@nameauth@Name: Enhanced         |    |
|      | \AKA: No local \newtoks .....      | 95 |      | workflow control .....            | 69 |
|      | nameauth: No local \newtoks ...    | 97 |      | \@nameauth@Parse: Enhanced,       |    |
| 2.5  |                                    |    |      | integrated caps .....             | 70 |
|      | \@nameauth@Hook: Improve hooks     | 74 |      | \@nameauth@UTFtest: The override  |    |
|      | \@nameauth@Name: Hooks query       |    |      | bypasses the test .....           | 67 |
|      | internal values .....              | 69 |      | \AKA: Can skip index .....        | 95 |
|      | \FrontNamesFormat: Added .....     | 65 |      | \AltCaps: Added .....             | 77 |
|      | General: No default formatting ... | 1  |      | \AltFormatActive: Added .....     | 77 |
| 2.6  |                                    |    |      | \AltFormatActive*: Added .....    | 77 |
|      | \@nameauth@Name: Improve           |    |      | \AltFormatInactive: Added ...     | 77 |
|      | indexing .....                     | 69 |      | \AltOff: Added .....              | 77 |
|      | \AKA: Fix index commas .....       | 95 |      | \AltOn: Added .....               | 77 |
|      | \IndexName: Fix commas .....       | 81 |      | \ForceName: Added .....           | 78 |
|      | \NoComma: Added .....              | 78 |      | \ForgetThis: Added .....          | 78 |
|      | General: Fix older syntax in most  |    |      | \IncludeName*: Fixed .....        | 87 |
|      | macros .....                       | 1  |      | \IndexName: Better tests .....    | 81 |
| 3.0  |                                    |    |      | \IndexRef: Better tests .....     | 82 |
|      | \@nameauth@Error: Added .....      | 68 |      | \JustIndex: Added .....           | 79 |
|      | \@nameauth@Hook: Better            |    |      | \KeepName: Added .....            | 78 |
|      | punctuation detection .....        | 74 |      | \NameParser: Older syntax fixed;  |    |
|      | \@nameauth@Name: Redesigned ..     | 69 |      | NBSP added .....                  | 79 |
|      | \@nameauth@NonWest: Added ...      | 72 |      | \NameQueryInfo: Short macro ..    | 90 |
|      | \@nameauth@Parse: Added .....      | 70 |      | \PName: Can skip index .....      | 96 |
|      | \@nameauth@TagRoot: Added ...      | 67 |      | \SkipIndex: Added .....           | 79 |
|      | \@nameauth@TrimRoot: Redesigned    | 67 |      | \SubvertName: Fix old syntax ..   | 94 |
|      | \@nameauth@TrimSuffix: New test    | 67 |      | \SubvertThis: Added .....         | 78 |
|      | \@nameauth@TrimTag: Added ...      | 67 |      | \textBF: Added .....              | 78 |
|      | \@nameauth@UTFtest: Added ...      | 67 |      | \textIT: Added .....              | 77 |
|      | \@nameauth@West: Added .....       | 73 |      | \textSC: Added .....              | 77 |
|      | \AKA: Redesigned .....             | 95 |      | \textUC: Added .....              | 77 |
|      | \DropAffix: Added .....            | 78 |      | General: Simplified logic and     |    |
|      | \ExcludeName: Redesigned .....     | 84 |      | argument tests in most macros     | 1  |
|      | \ForceFN: Added .....              | 76 |      |                                   |    |

## 5 Index

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

| Symbols                            |                                    |                                      |  |
|------------------------------------|------------------------------------|--------------------------------------|--|
| <code>\@nameauth@Actual</code>     | <a href="#">. 478</a>              | Atatürk                              | .....                                  |
| <code>\@nameauth@Cap</code>        | <a href="#">.... 117</a>           |                                      | <i>see</i> Kemal, Mustafa              |
| <code>\@nameauth@CheckDot</code>   | <a href="#">139</a>                | Attila the Hun                       | <a href="#">.... 8, 32</a>             |
| <code>\@nameauth@Cii</code>        | <a href="#">.... 123</a>           |                                      |  |
| <code>\@nameauth@Ciii</code>       | <a href="#">.... 125</a>           | <b>B</b>                             |  |
| <code>\@nameauth@Clean</code>      | <a href="#">... 88</a>             | Babbage, Charles                     | <a href="#">... 27</a>                 |
| <code>\@nameauth@Error</code>      | <a href="#">. 146</a>              | Bernard of Clairvaux                 | <a href="#">.. 45</a>                  |
| <code>\@nameauth@EvalDot</code>    | <a href="#">141</a>                | Bess, Good Queen                     | ...                                    |
| <code>\@nameauth@Hook</code>       | <a href="#">. 396</a>              |                                      | <i>see</i> Elizabeth I                 |
| <code>\@nameauth@Index</code>      | <a href="#">.. 444</a>             | Boëthius                             | <a href="#">..... 23</a>               |
| <code>\@nameauth@Name</code>       | <a href="#">... 159</a>            |                                      |  |
| <code>\@nameauth@NonWest</code>    | <a href="#">296</a>                | <b>C</b>                             |  |
| <code>\@nameauth@Parse</code>      | <a href="#">. 206</a>              | <code>\CapName</code>                | <a href="#">..... 20, 482</a>          |
| <code>\@nameauth@Root</code>       | <a href="#">.... 90</a>            | <code>\CapThis</code>                | <a href="#">..... 21, 479</a>          |
| <code>\@nameauth@Suffix</code>     | <a href="#">.. 94</a>              | Carnap, Rudolph                      | ...                                    |
| <code>\@nameauth@TagRoot</code>    | <a href="#">. 92</a>               |                                      | <a href="#">..... 24, 37, 40</a>       |
| <code>\@nameauth@TestDot</code>    | <a href="#">127</a>                | Carter, J.E., Jr., pres.             | .....                                  |
| <code>\@nameauth@TrimRoot</code>   | <a href="#">91</a>                 |                                      | <a href="#">..... 14, 41</a>           |
| <code>\@nameauth@TrimSuffix</code> | <a href="#">..... 95</a>           | Carter, Jimmy                        | <a href="#">.....</a>                  |
| <code>\@nameauth@TrimTag</code>    | <a href="#">. 93</a>               |                                      | <i>see</i> Carter, J.E., Jr.           |
| <code>\@nameauth@UTFTest</code>    | <a href="#">. 97</a>               | Chaplin, Charlie                     | <a href="#">..... 47</a>               |
| <code>\@nameauth@West</code>       | <a href="#">... 348</a>            | Chiang Kai-shek†, pres.              | <a href="#">..... 9, 21</a>            |
| <code>\@nameauth@toksa</code>      | <a href="#">.. 52</a>              | Cicero, M.T.                         | <a href="#">... 16, 17, 25</a>         |
| <code>\@nameauth@toksb</code>      | <a href="#">.. 52</a>              | Clemens, Samuel L.                   | ...                                    |
| <code>\@nameauth@toksc</code>      | <a href="#">.. 52</a>              |                                      | <i>see</i> Twain, Mark                 |
|                                    |                                    | Colfax, Schuyler, v.p.               | <a href="#">. 36</a>                   |
| <b>A</b>                           |                                    | Confucius                            | <a href="#">16, 17, 21, 25, 38</a>     |
| <code>\AccentCapThis</code>        | <a href="#">. 22, 480</a>          |                                      |  |
| ADAMS, John, pres.                 | <a href="#">55, 56</a>             | <b>D</b>                             |  |
| Æthelred II, king                  | ...                                | Dagobert I†, king                    | <a href="#">.... 9</a>                 |
|                                    | <a href="#">.... 7, 21, 23, 32</a> | DAVIS, Sammy, JR.                    | <a href="#">55, 56</a>                 |
| <code>\AKA</code>                  | <a href="#">..... 41, 1340</a>     | <code>[de Smet]</code> , Pierre-Jean | <a href="#">..... 57, 60</a>           |
| <code>\AKA*</code>                 | <a href="#">..... 41, 1386</a>     |                                      |  |
| <code>\AllCapsActive</code>        | <a href="#">. 20, 484</a>          | de Soto, Hernando                    | <a href="#">. 8, 21</a>                |
| <code>\AllCapsInactive</code>      | <a href="#">20, 483</a>            | Demetrius I Soter, king              | <a href="#">30</a>                     |
| <code>\AltCaps</code>              | <a href="#">..... 28, 517</a>      | <i>Doctor angelicus</i>              | ....                                   |
| <code>\AltFormatActive</code>      | <a href="#">26, 495</a>            |                                      | <i>see</i> Thomas Aquinas              |
| <code>\AltFormatActive*</code>     | <a href="#">..... 26, 499</a>      | <i>Doctor mellifluus</i>             | <i>see</i>                             |
| <code>\AltFormatInactive</code>    | <a href="#">..... 26, 503</a>      | Bernard of Clairvaux                 |  |
| <code>\AltOff</code>               | <a href="#">..... 28, 512</a>      | Dongen, Marc van                     | <a href="#">.. 2, 69</a>               |
| <code>\AltOn</code>                | <a href="#">..... 28, 507</a>      | <code>\DropAffix</code>              | <a href="#">.... 18, 533</a>           |
| Anthony, Susan B.                  | <a href="#">... 40</a>             | Du Bois, W.E.B.                      | <a href="#">..... 46</a>               |
| Arai Akino                         | <a href="#">..... 20</a>           | du Cange                             | <a href="#">..... see</a>              |
| Aristotle                          | <a href="#">..... 6, 8</a>         |                                      | <i>du Fresne, Charles</i>              |
| Arouet, François-Marie             | .....                              | du Fresne, Charles                   | <a href="#">... 42</a>                 |
|                                    | <i>see</i> Voltaire                | DuBois, W.E.B.                       | ....                                   |
|                                    |                                    |                                      | <i>see</i> Du Bois, W.E.B.             |
|                                    |                                    | <b>E</b>                             |  |
|                                    |                                    | Einstein, Albert                     | <a href="#">.... 16, 17, 25, 40</a>    |
|                                    |                                    | Elizabeth I, queen                   | ...                                    |
|                                    |                                    |                                      | <a href="#">..... 3, 6, 8,</a>         |
|                                    |                                    |                                      | <a href="#">16, 17, 25, 37, 42, 43</a> |
|                                    |                                    | environments:                        |  |
|                                    |                                    | nameauth                             | <a href="#">... 6, 1397</a>            |
|                                    |                                    | <code>\ExcludeName</code>            | <a href="#">.. 31, 801</a>             |
|                                    |                                    | <b>F</b>                             |  |
|                                    |                                    | <code>\FName</code>                  | <a href="#">..... 17, 635</a>          |
|                                    |                                    | <code>\FName*</code>                 | <a href="#">..... 17, 636</a>          |
|                                    |                                    | <code>\ForceFN</code>                | <a href="#">..... 17, 488</a>          |
|                                    |                                    | <code>\ForceName</code>              | <a href="#">.... 25, 540</a>           |
|                                    |                                    | <code>\ForgetName</code>             | <a href="#">.. 40, 1222</a>            |
|                                    |                                    | <code>\ForgetThis</code>             | <a href="#">... 40, 538</a>            |
|                                    |                                    | Friedrich I Barbarossa,              |  |
|                                    |                                    | emperor                              | <a href="#">..... 22</a>               |
|                                    |                                    | <code>\FrontNameHook</code>          | <a href="#">.. 24, 51</a>              |
|                                    |                                    | <code>\FrontNamesFormat</code>       | <a href="#">..... 24, 50</a>           |
|                                    |                                    | FUKUYAMA Takeshi                     | <a href="#">. 27</a>                   |
|                                    |                                    | <b>G</b>                             |  |
|                                    |                                    | GARBO, Greta                         | <a href="#">..... 27</a>               |
|                                    |                                    | <code>\GlobalNames</code>            | <a href="#">.. 40, 542</a>             |
|                                    |                                    | Goethe, J.W. von                     | ...                                    |
|                                    |                                    |                                      | <a href="#">..... 7, 18, 21, 30</a>    |
|                                    |                                    | Gossett, Louis, Jr.                  | <a href="#">.... 18</a>                |
|                                    |                                    | Grant, Ulysses S., pres.             | <a href="#">36</a>                     |
|                                    |                                    | Gregorio, Enrico                     | <a href="#">..... 2</a>                |
|                                    |                                    | Gregory I, pope                      | <a href="#">.. 34, 42</a>              |
|                                    |                                    | Gregory the Great                    | ...                                    |
|                                    |                                    |                                      | <i>see</i> Gregory I                   |
|                                    |                                    | <b>H</b>                             |  |
|                                    |                                    | Hammerstein, Oskar, II               | <a href="#">..... 18, 21</a>           |
|                                    |                                    | Harnack, Adolf                       | <a href="#">..... 30</a>               |
|                                    |                                    | Harun AL-RASHID                      | <a href="#">. 55, 56</a>               |
|                                    |                                    | Hearn, Lafcadio                      | <a href="#">..... 42</a>               |
|                                    |                                    | Henry VIII†, king                    | <a href="#">.... 9</a>                 |
|                                    |                                    | Hope, Bob                            | <a href="#">..... 37, 42</a>           |
|                                    |                                    | Hope, Leslie Townes                  | ...                                    |
|                                    |                                    |                                      | <i>see</i> Hope, Bob                   |
|                                    |                                    | <b>I</b>                             |  |
|                                    |                                    | <code>\if@nameauth@InAKA</code>      | <a href="#">52</a>                     |
|                                    |                                    | <code>\if@nameauth@InName</code>     | <a href="#">52</a>                     |
|                                    |                                    | <code>\IfAKA</code>                  | <a href="#">..... 38, 1187</a>         |

